TUGAS AKHIR – IF184802

PENGEMBANGAN *AD-HOC ON DEMAND DISTANCE VECTOR* (AODV) DENGAN KONSEP PROBABILITAS *CONNECTIVITY-AWARE* PADA *MOBILE AD-HOC NETWORK* (MANET)

MAULANA SECHAN
NRP 05111540000059

Dosen Pembimbing I
Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.

Dosen Pembimbing II
Ir. Muchammad Husni, M.Kom.

Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

(Halaman ini sengaja dikosongkan)

TUGAS AKHIR – IF184802

PENGEMBANGAN *AD-HOC ON DEMAND DISTANCE VECTOR* (AODV) DENGAN KONSEP PROBABILITAS *CONNECTIVITY-AWARE* PADA *MOBILE AD-HOC NETWORK* (MANET)

MAULANA SECHAN
NRP 05111540000059

Dosen Pembimbing I
Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.

Dosen Pembimbing II
Ir. Muchammad Husni, M.Kom.

Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESES – IF184802

DEVELOPMENT AD-HOC ON DEMAND DISTANCE VECTOR (AODV) WITH THE CONNECTIVITY-AWARE PROBABILITY CONCEPT IN MOBILE AD-HOC NETWORK (MANET)

MAULANA SECHAN
NRP 05111540000059

First Advisor
Dr.Eng. Radityo Anggoro, S.Kom, M.Sc.

Second Advisor
Ir. Muchammad Husni, M.Kom.

Department of Informatics Engineering
Faculty of Intelligent Electrical and Informatics Technology
Sepuluh Nopember Institute of Technology
Surabaya 2020

(Halaman ini sengaja dikosongkan)

PENGEMBANGAN *AD-HOC ON DEMAND DISTANCE VECTOR* (AODV) DENGAN KONSEP PROBABILITAS *CONNECTIVITY-AWARE* PADA *MOBILE AD-HOC NETWORK* (MANET)

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Arsitektur dan Jaringan Komputer
Program Studi S-1 Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh:
MAULANA SECHAN
NRP: 05111540000059

Disetujui oleh Pembimbing Tugas Akhir:

1. Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.
(NIP. 198410162008121002) ..... (Pembimbing 1)

2. Ir. Muchammad Husni, M.Kom.
(NIP. 196002211984031001) ..... (Pembimbing 2)

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN INSTITUT TEKNOLOGI SEPULUH NOPEMBER FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS DEPARTEMEN TEKNIK INFORMATIKA

SURABAYA
JANUARI, 2020

(Halaman ini sengaja dikosongkan)

PENGEMBANGAN *AD-HOC ON DEMAND DISTANCE VECTOR* (AODV) DENGAN KONSEP PROBABILITAS *CONNECTIVITY-AWARE* PADA *MOBILE AD-HOC NETWORK* (MANET)

Nama Mahasiswa	**: MAULANA SECHAN**
NRP	**: 05111540000059**
Departemen	**: Teknik Informatika FTEIC-ITS**
Dosen Pembimbing 1	**: Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**
Dosen Pembimbing 2	**: Ir. Muchammad Husni, M.Kom.**

Abstrak

Ad-hoc routing protocol memiliki tiga klasifikasi dalam penentuan rutenya yaitu *reactive protocol, proactive protocol,* dan *hybrid protocol.* AODV (*Ad-hoc On demand Distance Vector*) termasuk ke dalam *reactive protocol,* sebuah protokol yang hanya akan membuat rute ketika *node* sumber membutuhkannya. AODV memiliki dua fase, yaitu *route discovery* dan *route maintenance. Route discovery* digunakan untuk meminta dan meneruskan informasi rute hingga sampai ke destinasi yang terdiri dari proses pengiriman *Route Request* (RREQ) dan *Route Reply* (RREP). Sedangkan *route maintenance* digunakan untuk mengetahui informasi adanya kesalahan pada rute, jika terjadi kesalahan maka akan terjadi proses pengiriman *Route Error* (RERR).

Salah satu penerapan AODV yaitu pada lingkungan MANET (*Mobile Ad-hoc Network*). MANET sendiri merupakan sebuah teknologi WLAN (*Wireless Local Area Network*) yang tidak memerlukan infrastrukstur dalam jaringan sehingga mempermudah pemakai (*user*) dalam berkomunikasi dengan memanfaatkan keberadaan *mobile device* yang dimilikinya, sehingga sangat cocok diterapkan pada daerah yang memiliki kekurangan dalam hal infrastruktur telekomunikasi.

Tetapi pada saat penerapannya, topologi MANET sering berubah dengan cepat ketika *node* bergerak bebas tanpa batasan dalam hal arah atau mobilitas. Beberapa *node* yang memiliki kecepatan tinggi dan arah yang berbeda akan menyebabkan tautan nirkabel antar simpul-simpul yang ada sering mengalami kerusakan dan kadaluwarsa. Sedangkan untuk membangun kembali koneksi nirkabel memerlukan pembanjiran jaringan dengan sejumlah besar paket kontrol yang dapat mempengaruhi kinerja jaringan.

Pada Tugas Akhir ini, mengusulkan suatu solusi untuk menghadapi hal tersebut. Yaitu dengan skema probabilistik yang memanfaatkan vektor kecepatan pengirim dan penerima untuk menghitung sudut θ di antara mereka yang nantinya digunakan untuk mengatur probabilitas siaran ulang, serta ambang batas penghitung dengan waktu yang sesuai. Tujuan utama dari solusi ini adalah dengan memprioritaskan transmisi *node* dengan kecepatan yang sama untuk menjamin bahwa hanya node yang paling stabil yang berpartisipasi dalam fase penemuan rute demi menghindari fenomena pemutusan hubungan yang sering terjadi.

Dari hasil uji coba, AODV yang dimodifikasi pada skenario *node* yaitu skenario dengan jumlah *node* 20, 40 hingga 200 serta luas area uji 1000m x 1000m luas persegi dan kecepatan maksimum acak 20 m/s, serta skenario *speed* yaitu skenario dengan jumlah *node* 100 serta luas area uji 1000m x 1000m luas persegi dan kecepatan maksimum acak 5, 20 sampai 100 m/s. Pada skenario *node* berhasil meningkatkan nilai rata-rata *Packet Delivery Ratio* (PDR) hingga 10,67%, *End-To-End Delay* naik hingga 4,55% dan penurunan nilai rata-rata *Routing Overhead* (RO) hingga 21,8%.. Sedangkan pada skenario *speed* berhasil meningkatkan nilai rata-rata *Packet Delivery Ratio* (PDR) hingga 21,64%, menurunkan nilai *End-To-End Delay* hingga 11,2% dan penurunan nilai rata-rata *Routing Overhead* (RO) hingga 16,51%.

Kata kunci: ***MANETs, AODV, Probabilitas, Connectivity-aware, Unreliable Node, Reliable Node, Kecepatan***

DEVELOPMENT AD-HOC ON DEMAND DISTANCE VECTOR (AODV) WITH THE CONNECTIVITY-AWARE PROBABILITY CONCEPT IN MOBILE AD-HOC NETWORK (MANET)

Student's Name	**: MAULANA SECHAN**
Student's ID	**: 05111540000059**
Department	**: Informatics Engineering – FTEIC ITS**
First Advisor	**: Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**
Second Advisor	**: Ir. Muchammad Husni, M.Kom.**

Abstract

Ad-hoc routing protocol has three classifications in determining its route, namely reactive protocol, proactive protocol, and hybrid protocol. AODV (Ad-hoc On demand Distance Vector) is included in the reactive protocol, a protocol that will only route when the source node needs it. AODV has two phases, namely route discovery and route maintenance. Route discovery is used to request and forward route information up to the destination which consists of the process of sending Route Request (RREQ) and Route Reply (RREP). While route maintenance is used to find out information about an error on the route, if an error occurs, the sending error route (RERR) will occur.

One application of AODV is in the MANET (Mobile Ad-hoc Network) environment. MANET itself is a WLAN (Wireless Local Area Network) technology that does not require infrastructure in the network making it easier for users to communicate by utilizing the presence of their mobile devices, so it is very suitable to be applied in areas that have deficiencies in telecommunications infrastructure.

But at the time of its application, the MANET topology often changes rapidly when the node moves freely without restrictions in terms of direction or mobility. Some nodes that have high speeds and different directions will cause wireless links between existing nodes to often be damaged and expired. Meanwhile, to re-establish a wireless connection requires flooding the network with a large number of control packets that can affect network performance.

In this Final Project, proposes a solution to deal with this. Namely with a probabilistic scheme that utilizes the sender and receiver velocity vectors to calculate the angle between them which will be used to adjust the probability of rebroadcast, as well as the counter threshold with the appropriate time. The main objective of this solution is to prioritize the transmission of nodes at the same speed to ensure that only the most stable nodes participate in the route discovery phase in order to avoid the frequent termination phenomena.

From the test results, the AODV modified in the node scenario is a scenario with a number of nodes 20, 40 to 200 and a test area of 1000m x 1000m square area and a maximum random speed of 20 m / s, and a speed scenario that is a scenario with a number of node 100 and area test area of 1000m x 1000m square area and maximum random speed of 5, 20 to 100 m / s. In the scenario the node succeeded in increasing the average value of Packet Delivery Ratio (PDR) to 10.67%, End-To-End Delay increased to 4.55% and decreased the average value of Routing Overhead (RO) to 21.8%. Whereas the speed scenario succeeded in increasing the average value of Packet Delivery Ratio (PDR) to 21.64%, reducing the End-To-End Delay value by 11.2% and decreasing the average value of Routing Overhead (RO) to 16, 5%

Keyword: MANETs, AODV, Probability, Connectivity-aware, Unreliable Node, Reliable Node, Speed

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Puji syukur kepada Allah Yang Maha Esa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul **"Pengembangan*Ad-hoc On Demand Distance Vector* (AODV) Dengan Konsep Probabilitas *Connectivity-Aware* pada *Mobile Ad-hoc Network* (MANET)"**.

Harapan penulis semoga apa yang tertulis di dalam buku Tugas Akhir ini dapat bermanfaat bagi pengembangan ilmu pengetahuan saat ini dan ke depannya, serta dapat memberikan kontribusi yang nyata.

Dalam pelaksanaan dan pembuatan Tugas Akhir ini tentunya sangat banyak bantuan yang penulis terima dari berbagai pihak, tanpa mengurangi rasa hormat penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Allah SWT atas segala rahmat yang diberikan sehingga penulis dapat menyelesaikan Tugas Akhir ini.
2. Keluarga penulis terutama Bapak dan Ibu selaku orang tua penulis dan adik-adik penulis atas segala dukungan berupa motivasi, doa, moral, dan material sehingga penulis ttap semangat dan dapat menyelesaikan Tugas Akhir ini.
3. Bapak Dr. Eng. Radityo Anggoro, S.Kom., M.Sc., dan Bapak Ir. Muchammad Husni, M.Kom. selaku dosen pembimbing penulis atas nasihat, arahan dan bantuannya sehingga penulis dapat menyeleksaikan Tugas Akhir ini.
4. Teman-teman dari Penghuni Wardug (Haekal Azmi, Tegar, Pandito, Irvan, Faris, Pandu, Janar, Andi, Gandi, Ilham Penyok, Yuda, Budi, Cak Tek, Cak Pentol, Mas Atam, Mbak Diana dan Nabil) yang selalu memberikan semangat, hiburan, dan menjadi tempat bertukar pikiram dan pendapat serta menemani penulis sehari-hari di ITS selama penulis berkuliah di Departemen Teknik Informatika ITS.

5. Teman-teman dari keluarga besar Laboratorium AJK (Ilham Penyok, Fuad, Didin, Awan, Satriya, Daus, Nahda, dan Hana), Laboratorium MI (Huda, Narendra, Ivan, Adib, Yudhis, Rezky, Unggul, Yola, Bela, Salma, Ajeng, Nafi, Dio, Subhan, Yuga, Zahri, Byan, Adi, GD, dan Azzam), serta teman-teman Informatika angkatan 2015 yang telah menemani, memotivasi, memberikan doa, memberikan hiburan di kala penulis sedang jenuh saat pengerjaan Tugas Akhir ini.
6. Teman-teman BEM ITS Gelora Aksi (Mas Krisna, Mas Ihram, Nandya, Alfa, Yufa, Vicario, Bagus, Fara, Alvian, Boy, Hanny, Nisa, Tika, Achmad, Noel, Sat, Aryo, Aang, Nug, Sasha, Syna, Audi, Hadi, Icha, Iid, Kinan, Marko, Nurul, Rizka dan Azary) sudah memberikan kesibukan lebih untuk penulis semasa kuliah di Informatika ITS.
7. Teman-teman Gerigi ITS, HMTC Inspirasi dan HMTC Kreasi yang sudah memberikan kesibukan lebih untuk penulis semasa kuliah di Informatika ITS.
8. Teman-teman SMA penulis (Vioza, Kevin, Bima) yang telah memberikan hiburan di kala penulis sedang jenuh saat pengerjaan Tugas Akhir ini.

Penulis telah berusaha sebaik-baiknya dalam menyusun Tugas Akhir ini, namun penulis menyadari bahwa masih terdapat kekurangan, kesalahan, maupun kelalaian yang telah penulis lakukan. Oleh karena itu, saran dan kritik yang membangun sangat dibutuhkan untuk penyempurnaan Tugas Akhir ini.

Surabaya, Desember 2019

MAULANA SECHAN

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Saat ini perkembangan teknologi telah memiliki kemajuan yang sangat pesat sehingga dapat mencapat titik dimana teknologi dapat menghubungkan antar manusia untuk saling berkomunikasi. Untuk saling berkomunikasi secara *online*, teknologi membutuhkan suatu *network* yang saling terhubung. *Network* sendiri memiliki fungsi untuk *transfer* data, dengan kata lain berkomunikasi secara *online* merupakan suatu proses pertukaran data antar pengguna. Salah satu sistem *network* yang sedang banyak digunakan dan dikembangkan adalah MANET (*Mobile Ad-hoc Network*). MANET sendiri merupakan sebuah teknologi WLAN (*Wireless Local Area Network*) yang tidak memerlukan infrastrukstur dalam jaringan sehingga mempermudah pemakai (*user*) dalam berkomunikasi dengan memanfaatkan keberadaan *mobile device* yang dimilikinya. Itu juga yang membuat MANET sangat cocok diterapkan pada daerah yang memiliki kekurangan dalam hal infrastruktur telekomunikasi.

Selanjutnya pada saat pertukaran data sedang berlangsung, kita juga perlu memastikan agar setiap *user/node* memiliki suatu *link* yang saling terhubung. Cara untuk menghubungkan setiap *node* agar saling terhubung merupakan proses *routing*. AODV (*Ad-hoc On demand Distance Vector*) merupakan salah satu cara *routing* yang merupakan *reactif protocol*, yang berarti *routing* akan terbentuk jika terdapat permintaan.

Tetapi pada saat penerapannya, topologi MANET sering berubah dengan cepat ketika *node* bergerak bebas tanpa batasan dalam hal arah atau mobilitas. Maka yang terjadi perutean data dan penyebaran paket akan menjadi tugas yang menantang. Ini disebabkan karena biasanya untuk *node* yang memiliki kecepatan

tinggi dan arah yang berbeda akan menyebabkan tautan nirkabel antar simpul-simpul yang ada sering mengalami kerusakan dan sering kadaluwarsa. Sedangkan untuk membangun kembali koneksi nirkabel memerlukan pembanjiran jaringan dengan sejumlah besar paket kontrol seperti *Route Replay Packets* (RREP) dan *Route Error Packets* (RERR), di samping paket-paket tambahan RREQ. Misalnya, dalam protokol AODV, fase penemuan rute menukar jaringan dengan paket kontrol RREQ untuk menemukan rute optimal ke tujuan yang diperlukan. Dalam beberapa kasus, rute yang ditetapkan dapat berisi node yang tidak stabil, di mana kerusakan tautan sering terjadi dan mempengaruhi kinerja jaringan secara keseluruhan.

Pada Tugas Akhir ini, mengusulkan suatu solusi untuk menghadapi hal tersebut. Yaitu dengan skema probabilistik yang memanfaatkan vektor kecepatan pengirim dan penerima untuk menghitung sudut θ di antara mereka yang nantinya digunakan untuk mengatur probabilitas siaran ulang, serta ambang batas penghitung dengan waktu yang sesuai. Tujuan utama dari solusi ini adalah dengan memprioritaskan transmisi *node* dengan kecepatan yang sama untuk menjamin bahwa hanya *node* yang paling stabil yang berpartisipasi dalam fase penemuan rute demi menghindari fenomena pemutusan hubungan yang sering terjadi.

1.2 Rumusan Masalah

Tugas Akhir ini mengangkat beberapa rumusan masalah sebagai berikut:

1. Bagaimana cara membagi *node* menjadi 2 jenis, yaitu *reliable node* dan *unreliable node* hingga membuat semua rute antar *node* yang terbentuk hanya dibentuk oleh *reliable node* saja?
2. Bagaimana dampak penerapan modifikasi terhadap performa AODV pada lingkungan MANET?

1.3 Batasan Permasalahan

Permasalahan yang dibahas pada Tugas Akhir ini memiliki batasan sebagai berikut:

1. Protokol jaringan yang digunakan adalah WLAN (*Wireless Local Area Network*) pada jaringan *Mobile Ad-hoc Network* (MANET).
2. Dasar dari *routing protocol* yang akan diujicobakan adalah AODV.
3. Simulasi pembuatan dan pengujian skenario uji coba jaringan menggunakan *Network Simulator* 2 (NS-2).

1.4 Tujuan

Tujuan dari Tugas Akhir ini adalah sebagai berikut:

1. Menentukan setiap *node* apakah *node* tersebut termasuk *reliable node* atau *unreliable node* hingga membuat rute yang terbentuk hanya dibentuk oleh *reliable node* saja, agar semua *link* yang tersampaikan oleh semua *node* tidak rusak dan tidak membanjiri jaringan.
2. Menganalisa performa AODV yang telah dimodifikasi berdasarkan matriks *Packet Delivery Ratio, End-to-end Delay*, dan *Routing Overhead*.

1.5 Manfaat

Manfaat yang diperoleh dari pengerjaan Tugas Akhir ini adalah dapat mengembangkan *route discovery* pada *protocol routing* AODV yang diterapkan pada sistem network MANET berdasarkan skema probabilistik kecepatan pada setiap *node* agar dapat menangani beberapa kendala yang terjadi pada saat fase penemuan rute sedang berlangsung. Seperti, banyaknya *link* yang rusak dan kadaluarsa serta kebanjiran jaringan yang nantinya dapat mengganggu sistem *network* yang ada.

1.6 Metodologi

Pembuatan Tugas Akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal Tugas Akhir

Tahapan awal dari Tugas Akhir ini adalah penyusunan Proposal Tugas Akhir. Proposal Tugas Akhir berisi pendahuluan, deskripsi dan gagasan metode-metode yang dibuat dalam Tugas Akhir ini. Pendahuluan ini terdiri atas hal yang menjadi latar belakang diajukannya Tugas Akhir, rumusan masalah yang diangkat, batasan masalah untuk Tugas Akhir, dan manfaat dari hasil pembuatan Tugas Akhir ini. Selain itu dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan Tugas Akhir.

1.6.2 Studi Literatur

Pada tahap ini, dipelajari sejumlah referensi yang diperlukan dalam melakukan implementasi yaitu mengenai MANET, AODV, *Network Simulator* NS2 dan AWK.

1.6.3 Analisis dan Desain Sistem

Pada tahap ini dilakukan analisis dari hasil percobaan modifikasi AODV yang dibuat. Data yang dianalis berasal dari perhitungan *Packet Delivery Ratio*, *Routing Overhead*, *Forwarded Route Request*, dan *End-to-End Delay* paket dari *node* ke *node* lainnya. Hal ini bertujuan untuk merumuskan solusi yang tepat untuk konfigurasi AODV yang dimodifikasi dalam lingkungan topologi MANET.

1.6.4 Implementasi Sistem

Implementasi merupakan tahap untuk membangun metode-metode yang sudah diajukan pada proposal Tugas Akhir. Pada tahap ini dilakukan implementasi menggunakan NS-2 sebagai *simulator*, Bahasa C/C++ sebagai bahasa pemrograman, dan sebagai *tools* untuk uji coba dan mengimplentasikan desain sistem yang sudah dirancang.

1.6.5 Pengujian dan Evaluasi

Pada tahap ini dilakukan pengujian dengan membuat skenario untuk jumlah *node* yang terus ditambah sebagai pembanding juga skenario untuk kecepatan maksimum acak yang terus bertambah guna menghasilkan keadaan yang diujikan. Hasil dari pembuatan skenario tersebut akan dijalankan pada NS-2 sehingga menghasilkan *trace file*. Dari *trace file* tersebut beberapa poin-poin, seperti *Packet Delivery Ratio*, *End-to-end Delay*, *Routing Overhead*, dan *Forwarded Route Request* akan dihitung melalui *AWK Script* untuk mendapatkan hasil uji performa AODV yang telah dimodifikasi.

1.6.6 Penyusunan Buku

Pada tahap ini dilakukan penyusunan buku yang menjelaskan seluruh konsep, teori dasar dari metode yang digunakan, implementasi, serta hasil yang telah dikerjakan sebagai dokumentasi dari pelaksanaan Tugas Akhir.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan Tugas Akhir adalah sebagai berikut:

1. Bab I. Pendahuluan

Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.

2. Bab II. Tinjauan Pustaka
   Bab ini berisi kajian teori atau penjelasan dari metode, algoritma, *library*, dan *tools* yang digunakan dalam penyusunan Tugas Akhir ini. Bab ini berisi tentang penjelasan singkat mengenai MANET, AODV, NS2 dan AWK.
3. Bab III. Perancangan
   Bab ini berisi pembahasan mengenai perancangan skenario mobilitas, perancangan simulasi pada NS2, perancangan modifikasi AODV, serta perancangan metrik analisis (*Packet Delivery Ratio, End-to-end Delay, Routing Overhead*, dan *Forwarded Route Request.*).
4. Bab IV. Implementasi
   Bab ini menjelaskan implementasi yang berbentuk kode sumber dari proses modifikasi protokol AODV, pembuatan simulasi pada NS2, dan perhitungan metrik analisis.
5. Bab V. Uji Coba dan Evaluasi
   Bab ini berisikan hasil uji coba dan evaluasi dari implementasi yang telah dilakukan untuk menyelesaikan masalah yang dibahas pada Tugas Akhir.
6. Bab VI. Kesimpulan dan Saran
   Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.
7. Daftar Pustaka
   Bab ini berisi daftar pustaka yang dijadikan literatur dalam Tugas Akhir.
8. Lampiran
   Dalam lampiran terdapat tabel-tabel data hasil uji coba dan kode sumber program secara keseluruhan.

BAB II
TINJAUAN PUSTAKA

Bab ini berisi pembahasan mengenai teori-teori dasar atau penjelasan dari metode dan *tools* yang digunakan dalam Tugas Akhir.

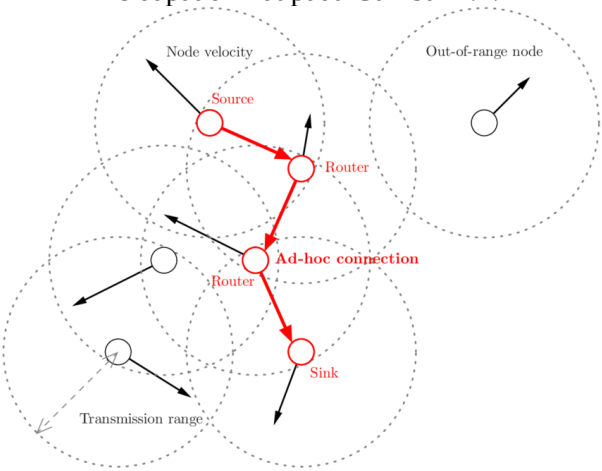
2.1 MANET

Mobile Ad-hoc Network atau MANET merupakan sebuah jaringan yang tidak menggunakan infrastruktur permanen. Hal ini disebabkan karena tingkat mobilitas *node* yang tinggi menyebabkan MANET tidak memerlukan infrastruktur yang seperti layaknya topologi jaringan yang lain. MANET merupakan jaringan *wireless* yang berbasis IEEE 802.11. Secara singkat MANET merupakan sistem jaringan dimana setiap *node* tidak hanya berperan sebagai *host* atau perangkat pengguna saja, akan tetapi setiap *node* berperan sebagai *router* yang dapat meneruskan paket kepada *node* lain yang menjadi tujuan pengiriman paket tersebut [12].

MANET memiliki sistem administrasi jaringan yang tidak terpusat, setiap *node* dapat keluar-masuk dari jaringan dengan mudah. Kemampuan *node* tidak hanya sebatas mengirim dan menerima data, akan tetapi *node* mampu untuk menjadi pengubung antar *node* lain [14]. Jangkauan transmisi dari media *wireless* setiap *node* tidak selalu lebar, sehingga ketika *node* sumber ingin berhubungan dengan *node* tujuan, maka diperlukan *node* lain yang menjembatani koneksi antar *node* tersebut [14]. Kondisi *node* yang mobile menyebabkan topologi pada jaringan MANET berubah dengan cepat. *Node* yang keluar dari jaringan akan menyebabkan putusnya jalur pengiriman paket data yang sudah ada. Ketika kondisi itu terjadi pada saat pengiriman paket, jalur lain harus segera ditemukan sehingga paket dapat sampai pada tujuan [15]. Oleh karena itu, protokol *routing* sangat

diperlukan untuk menentukan rute atau jalur pada jaringan MANET.

Protokol *routing* pada jaringan MANET diklasifikasikan menjadi tiga kelas menurut performansi dan fungsionalitasnya, yaitu: *tabledriven (proactive), on-deman (reactive),* dan *hybrid*. Setiap jenis protokol routing memiliki karakteristik kelebihan dan kekurangan masingmasing. Protokol *routing proactive* adalah protokol *routing* pada MANET yang melakukan *broadcast* secara terus-menerus untuk mengorganisir tabel *routing* pada setiap *node*, karena perubahan topologi jaringan yang dinamis. Kelebihan protokol *routing reactive* adalah penentuan jalur *routing* pada *node* hanya dilakukan pada saat dibutuhkan saja. Protokol *routing hybrid* adalah jenis protokol routing yang dalam penentuan jalurnya, memanfaatkan fungsional dari gabungan *proactive* dan *reactive* protokol *routing*. Tetapi pada saat penerapannya *node* akan bergerak bebas tanpa batasan dalam hal arah atau mobilitas. Perubahaan pergerakan inilah yang menjadi salah satu permasalahan dalam pengiriman paket data sehingga dibutuhkan informasi jarak antar *node*, kecepatan dan *delay* transmisi [3]. Ilustrasi MANETs dapat dilihat pada Gambar 2.1.



Gambar 2.1 Ilustrasi MANET

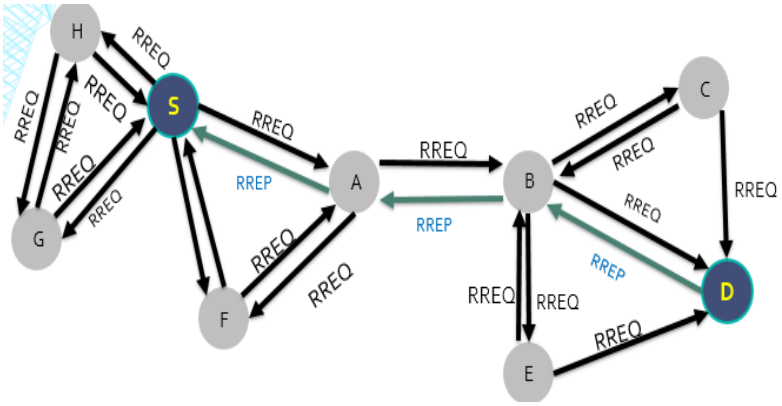
Dalam Tugas Akhir ini, penulis akan mengimplementasikan *routing protocol* AODV yang dimodifikasi dan menguji performa protokol tersebut pada lingkungan MANETs.

2.2 *Ad-hoc On demand Distance Vector* (AODV)

Ad-hoc On demand Distance Vector (AODV) adalah salah satu *routing* protokol yang dirancang untuk jaringan *ad-hoc mobile* dan termasuk dalam klasifikasi *reactive routing protocol*. Dimana merupakan algoritman *routing* permintaan, yang berarti sebuah protokol yang hanya membuat sebuah rute antara node hanya saat dibutuhkan. AODV menggunakan table routing satu *entry* untuk setiap tujuan, tanpa menggunakan routing.

Ada dua tahapan dalam AODV yaitu *route discovery* dan r*oute maintenance. Route discovery* memiliki dua pesan yaitu berupa *Route Request* (RREQ) dan *Route Reply* (RREP). Sedangkan *Route maintenance* berupa *Route Error* (RERR). AODV mempercayakan pada tabel *routing* untuk menyebarkan *Route Reply* kembali ke sumber dan mengarahkan paket menuju tujuan. Ciri utama dari AODV adalah menjaga *timer-based state* pada setiap *node* sesuai dengan penggunaan tabel *routing*.

AODV adalah sebuah metode *routing* pesan antar *node* yang memungkinkan *node-node* tersebut untuk melewatkan pesan melalui lingkungannya ke *node* yang tidak dapat dihubungi secara langsung. AODV melakukan ini dengan cara menemukan rute yang bisa dilalui oleh pesan. Selain itu AODV juga memastikan rute ini tidak mengandung perulangan (*loop*), menangani perubahan rute, dan membuat rute baru apabila terjadi *error* [4]. Ilustrasi pencarian rute oleh AODV dapat dilihat pada Gambar 2.2. Struktur paket RREQ dapat dilihat pada Tabel 2.1.



Gambar 2.2 Ilustrasi Pencarian Rute *Routing Protocol* AODV *[3]*

Tabel 2.1 Struktur Paket RREQ

source_addr	*source_sequence_#*	*broadcast_id*
dest_addr	*dest_sequence_#*	*hop_cnt*

Pada setiap *node* yang menggunakan protokol AODV pasti memiliki sebuah *routing table* dengan *field* sebagai berikut:

- *Destination Address:* berisi alamat dari *node* tujuan.
- *Destination Sequence Number*: *sequence number* dari jalur komunikasi.
- *Next Hop*: alamat *node* yang akan meneruskan paket data.
- *Hop Count*: jumlah *hop* yang harus dilakukan agar paket dapat mencapai *node* tujuan.
- *Lifetime*: waktu dalam milidetik yang diperlukan *node* untuk menerima RREP.
- *Routing Flags*: status jalur. Terdapat tiga tipe status, yaitu *up* (valid), *down* (tidak valid) atau sedang diperbaiki.

Sebagai contoh proses *route discovery* dalam AODV, ilustrasi pada Gambar 2.2 menggambarkan bagaimana *source node*, yaitu *node* berwarna kuning mencari rute untuk menuju *destination node* yaitu *node* berwarna merah. *Node* S akan membuat paket RREQ dan melakukan *broadcast* kepada semua *node* θngganya (*neighbor node*). Jika *destination sequence number* yang terdapat pada paket RREQ sama atau lebih kecil dari yang ada pada *routing table* dan rute menuju *node* tujuan belum ditemukan, maka paket tersebut tidak akan dilanjutkan (*drop*). Jika *destination sequence number* pada RREQ lebih besar dibandingkan dengan yang terdapat pada *routing table*, maka *entry* pada *routing table* akan diperbarui dan paket tersebut akan diteruskan oleh *neighbor node* sekaligus membuat *reverse path* menuju *source node.* Paket RREQ akan diteruskan hingga mencapai *node* D. Kemudian, jika rute menuju *node* D sudah terbentuk di dalam *routing table* dan memiliki *routing flags* "*up*", maka *node* D akan mengirimkan paket RREP melalui rute tersebut menuju *node*.

2.3 *Network Simulator-2* (NS-2)

Network Simulator 2 (NS-2) merupakan sebuah network simulator yang dibuat dengan tujuan riset dan pendidikan. Pada awalnya, NS dibangun sebagai varian dari *Real Network Simulator* pada tahun 1989 di University of California Berkeley dan USC ISI sebagai bagian dari projek *Virtual INternet Testbed* (VINT). NS yang banyak dikenal dengan NS-2 (versi 2) menjadi salah satu *tool* yang sangat berguna untuk menunjukkan simulasi jaringan melibatkan *Local Area Network*, *Wide Area Network* (WAN), dan telah berkembang selama beberapa tahun belakangan untuk memasukkan jaringan nirkabel (*wireless*) dan juga jaringan *ad hoc.* NS-2 memiliki beberapa fitur kelebihan yang dapat dimanfaatkan dalam pemodelan dan pengujian MANET.

Pada Tugas Akhir ini, NS-2 digunakan untuk melakukan simulasi lingkungan MANETs menggunakan protokol AODV yang sudah dimodifikasi. *Trace file* yang dihasilkan oleh NS-2

untuk mengukur peforma *routing* protokol AODV yang dimodifikasi.

2.3.1 Instalasi

NS-2 membutuhkan beberapa *package* yang harus sudah ter*install* sebelum memulai instalasi NS-2. Untuk meng*install dependency* yang dibutuhkan dapat dilakukan dengan *command* yang ditunjukkan pada Gambar 2.3.

```
sudo apt-get install build-essential automake
autoconf libxmu-dev
```

Gambar 2.3 Perintah untuk Meng-*install Dependency* NS-2

Setelah meng*install dependency* yang dibutuhkan, ekstrak *package* NS-2 dan ubah baris kode ke-137 pada *file* ls.h di *folder* linkstate menjadi seperti pada Gambar 2.4.

```
void eraseAll() {this->erase(baseMap::begin(),
baseMap::end()); }
```

Gambar 2.4 Baris kode yang diubah pada file ls.h

2.3.2 *Trace File*

Trace file merupakan *file* hasil simulasi yang dilakukan oleh NS-2 dan berisikan informasi detail pengiriman paket data. *Trace file* digunakan untuk menganalisis performa *routing protocol* yang disimulasikan. Detail penjelasan *trace file* ditunjukkan pada Tabel 2.2.

Tabel 2.2 Detail Penjelasan *Trace File* AODV

Kolom ke-	Penjelasan	Isi
1	*Event*	s : *sent*

		r : *received* f : *forwarded* D : *dropped*
2	*Time*	Waktu terjadinya *event*
3	ID *Node*	_x_ : dari 0 hingga banyak *node* pada topologi
4	*Layer*	AGT : *application* RTR : *routing* LL : *link layer* IFQ : *packet queue* MAC : *MAC* PHY : *physical*
5	*Flag*	--- : Tidak ada
6	*Sequence Number*	Nomor paket
7	*Packet Type*	AODV : paket *routing* AODV cbr : berkas paket CBR (*Constant Bit Rate*) RTS : *Request To Send* yang dihasilkan MAC 802.11 CTS : *Clear To Send* yang dihasilkan MAC 802.11 ACK : MAC ACK ARP : Paket *link layer address resolution protocol*
8	Ukuran	Ukuran paket pada *layer* saat itu
9	Detail MAC	[a b c d] a : perkiraan waktu paket b : alamat penerima c : alamat penerima d : IP header
10	*Flag*	-------- : Tidak ada
11	*Detail* IP *source, destination,* dan *nexthop*	[a:b c:d e f] a : IP *source node* b : *port source node*

		c : IP *destination node* (jika -1 berarti *broadcast*) d : *port destination node* e : IP *header* ttl f : IP *nexthop* (jika 0 berarti *node* 0 atau *broadcast*)

2.4 AWK

AWK adalah bahasa pemrograman yang digunakan untuk melakukan *text processing* dan ekstraksi data [6]. AWK merupakan sebuah program filter untuk teks, seperti halnya perintah grep pada terminal linux. AWK dapat digunakan untuk mencari bentuk atau model dalam sebuah berkas teks ke dalam bentuk teks lain. AWK dapat juga digunakan untuk melakukan proses aritmatika seperti yang dilakukan oleh perintah expr. AWK sama halnya seperti bahasa shell atau C yang memiliki karakteristik yaitu sebagai *tool* yang cocok untuk *jobs* juga sebagai pelengkap untuk *filter* standar.

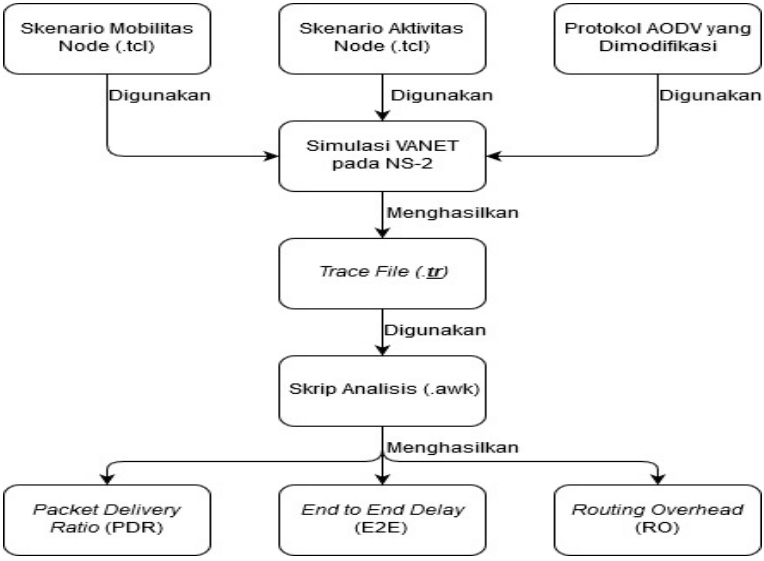
Pada Tugas Akhir ini, AWK digunakan untuk membuat script menghitung *Packet Delivery Ratio (*PDR*), End-to-end Delay, Routing Overhead* (RO) dari *trace file* NS2.

BAB III
PERANCANGAN

Perancangan merupakan bagian penting dari pembuatan sistem secara teknis sehingga bab ini secara khusus menjelaskan perancangan sistem yang dibuat dalam Tugas Akhir. Berawal dari deskripsi umum sistem hingga perancangan skenario, alur dan implementasinya.

3.1 Deskripsi Umum

Pada Tugas Akhir ini akan diimplementasikan *routing protocol* AODV dengan memodifikasi pada bagian proses *route discovery* yang dijalankan pada simulator NS-2. Diagram dari rancangan simulasi dari AODV asli dan AODV modifikasi dapat dilihat pada Gambar 3.1.



Gambar 3.1 Diagram Alur Rancangan Simulasi AODV Modifikasi

Modifikasi akan diawali dengan mendapatkan kecepatan dari *node* pengirim RREQ juga *node-node* θngga yang menerima RREQ. Setelah kecepatan didapatkan kita akan membagi semua node yang ada menjadi 2 jenis, yaitu *reliable node* dan *unreliable node*. *Reliable node* adalah *node* yang berhak untuk mengirim ulang / *rebroadcast* RREQ yang diterimanya. Sedangkan *Unreliable node* adalah *node* yang tidak berhak untuk mengirim ulang / *rebroadcast* RREQ yang diterimanya. Cara menentukan apakah *node* tersebut *reliable* atau *unreliable* adalah dengan membandingkan *threshold* yang ada dengan sudut teta (θ) yang terbentuk antara kecepatan *node* pengirim dengan *node* penerima. Sudut θ itu akan merepresentasikan kemiripan kecepatan antara pengirim RREQ dan penerima RREQ. Semakin kecil sudut θ semakin mirip kecepatannya dan semakin aman jika dibentuk suatu rute melaluinya. Dan ketika sudut θ kurang dari *threshold* yang sudah ditentukan maka dia akan dianggap sebagai *reliable node* dan dapat mem-*broadcast* ulang RREQ yang sudah diterimanya, namun ketika sudut θ lebih dari *threshold* yang sudah ditentukan maka dia dianggap sebagai *unreliable node* sehingga tidak dapat meneruskan RREQ yang sudah diterimanya.

Modifikasi yang telah dilakukan akan disimulasikan pada NS-2 di skenario dengan jumlah *node* yang terus bertambah sebagai pembanding juga skenario dengan kecepatan maksimum acak yang terus bertambah sebagai pembanding. Simulasi tersebut akan memberikan hasil *trace file* yang kemudian dianalisis menggunakan skrip AWK untuk mendapatkan *Packet Delivery Ratio* (PDR), *End-to-end Delay (*E2E) dan *Routing Overhead* (RO). Analisis tersebut dapat mengukur performa *routing protocol* AODV yang telah dimodifikasi dibandingkan dengan AODV sebelum dimodifikasi. Analisis ini digunakan untuk mengukur tingkat reliabilitas pengiriman data antara protokol AODV dengan

protokol AODV yang dimodifikasi. Daftar istilah yang sering digunakan pada buku Tugas Akhir ini dapat dilihat pada Tabel 3.1.

Tabel 3.1 Daftar Istilah

No.	Istilah	Penjelasan
1	AODV	Singkatan dari *Ad hoc On-demand Distance Vector*. Protokol yang digunakan pada Tugas Akhir ini.
2	PDR	*Packet Delivery Ratio*. Salah satu metrik analisis yang diukur. Berupa rasio jumlah pengiriman paket yang terkirim.
3	E2E	*Average End-to-End Delay*. Jeda waktu yang diukur saat paket terkirim.
4	RO	*Routing Overhead*. Jumlah *control packet* yang terkirim
5	RREQ	*Route Request*. Paket *request* pada AODV yang dikirim untuk mendapatkan rute.
6	RREP	*Route Reply*. Paket *reply* pada AODV yang dikirim ke *node* sumber melalui rute yang sudah terbuat.
7	*Reliable Node*	*Node* yang diperbolehkan untuk mengirimkan paket RREQ
8	*Unreliable Node*	*Node* yang tidak diperbolehkan untuk mengirimkan paket RREQ
9	*Threshold*	Batas nilai yang dijadikan sebagai acuan

3.2 Perancangan Skenario Mobilitas

Perancangan skenario mobilitas dimulai dengan membuat area simulasi, pergerakan *node*, dan implementasi pergerakan *node*. Dalam Tugas Akhir ini, terdapat dua macam area simulasi yang akan digunakan yaitu skenario *node* serta skenario *speed*. Skenario *node* adalah skenario dengan jumlah *node* yang terus bertambah sebagai pembanding. Skenario ini didapatkan dengan

jumlah *node* 20, 40 yang bertambah hingga 200 serta luas area uji 1000m x 1000m luas persegi dan kecepatan maksimum acak 20 m/s yang nantinya hasil dari skenario tersebut akan menjadi pembanding antara aodv modifikasi dengan aodv sebelum modifikasi. Sedangkan skenario *speed* adalah skenario dengan kecepatan maksimum acak yang terus bertambah sebagai pembanding. Skenario ini didapatkan dengan jumlah *node* 100 serta luas area uji 1000m x 1000m luas persegi dan kecepatan maksimum acak 5, 20 yang terus bertambah sampai 100 m/s yang nantinya hasil dari skenario tersebut akan menjadi pembanding antara aodv modifikasi dengan aodv sebelum modifikasi.

3.2.1 Perancangan Skenario *Threshold* Terbaik

Perancangan skenario mobilitas *threshold* terbaik diawali dengan merancang luas area skenario yang dibutuhkan. Luas area tersebut bisa ditentukan secara langsung sesuai dengan *paper* yang diteliti yaitu 1000m x 1000m.

Skenario *threshold* terbaik yang telah ditentukan luasnya juga memerlukan pengaturan jumlah node sebesar 100 *node* serta memerlukan pengaturan kecepatan maksimum acak sebesar 20 m/s. Skenario *threshold* terbaik kemudian dibuat dengan menggunakan NS-2 dan akan memiliki ekstensi .tcl.

Dari sinilah skenario mobilitas *threshold* dihasilkan berupa file .tcl yang berisi sebuah skenario dengan jumlah *node*, kecepatan maksimum acak dan luas area yang sudah ditentukan diawal dengan peletakan dan pergerakan *node* yang acak. Selain itu pada skenario ini kita juga memodifikasi AODV yang sudah kita modifikasi dengan mengatur rentang *threshold* yang diperlukan AODV modifikasi pada 0º, 33º hingga 165º.

3.2.2 Perancangan Skenario *Node*

Perancangan skenario mobilitas *node* diawali dengan merancang luas area skenario *node* yang dibutuhkan. Luas area

tersebut bisa ditentukan secara langsung sesuai dengan *paper* yang diteliti yaitu 1000m x 1000m.

Skenario *node* yang telah ditentukan luasnya juga memerlukan pengaturan kecepatan maksimum acak sebesar 20 m/s serta memerlukan pengaturan jumlah node yang terus bertambah dengan rentang 20, 40 hingga 200 *node* perskenario. Skenario *node* kemudian dibuat dengan menggunakan NS-2 dan akan memiliki ekstensi .tcl.

Dari sinilah skenario mobilitas *node* dihasilkan berupa file .tcl yang berisi sebuah skenario dengan jumlah *node* yang terus bertambah sesuai rentang yang diberikan, kecepatan maksimum acak dan luas area yang sudah ditentukan diawal dengan peletakan dan pergerakan *node* yang acak.

3.2.3 Perancangan Skenario *Speed*

Perancangan skenario mobilitas *speed* diawali dengan merancang luas area skenario *speed* yang dibutuhkan. Luas area tersebut bisa ditentukan secara langsung sesuai dengan *paper* yang diteliti yaitu 1000m x 1000m.

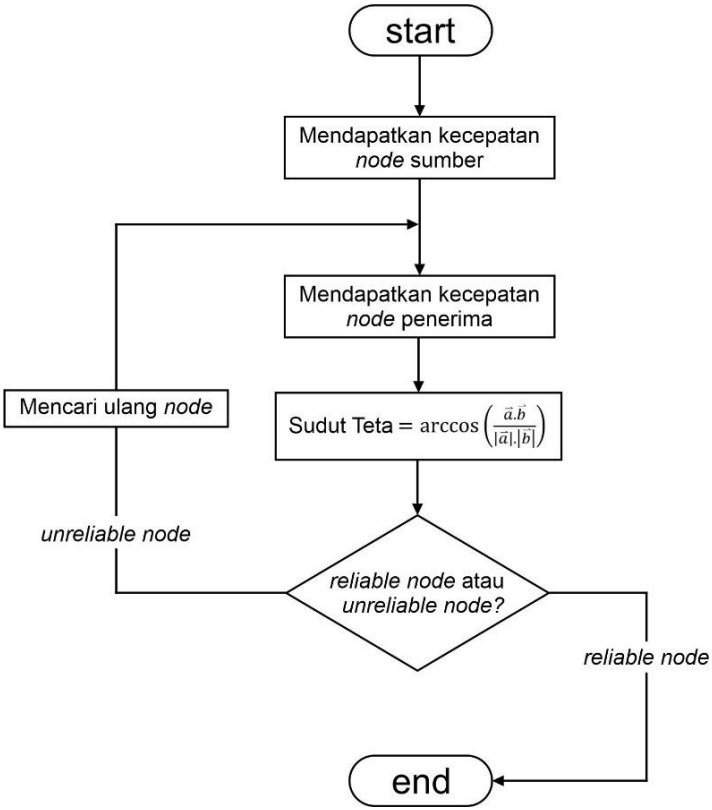
Skenario *speed* yang telah ditentukan luasnya juga memerlukan pengaturan jumlah node sebesar 100 *node* serta memerlukan pengaturan kecepatan maksimum acak yang terus bertambah dengan rentang 5 m/s, 20 m./s hingga 100 m/s perskenario. Skenario *speed* kemudian dibuat dengan menggunakan NS-2 dan akan memiliki ekstensi .tcl.

Dari sinilah skenario mobilitas *speed* dihasilkan berupa file .tcl yang berisi sebuah skenario dengan jumlah *node*, kecepatan maksimum acak yang terus bertambah sesuai rentang yang diberikan dan luas area yang sudah ditentukan diawal dengan peletakan dan pergerakan *node* yang acak.

3.3 Perancangan Modifikasi *Routing Protocol* AODV

Protokol AODV yang diajukan pada Tugas Akhir ini merupakan modifikasi dari protokol AODV dengan mengubah mekanisme *route discovery* pada protokol tersebut. Pada Gambar 3.2 dijelaskan alur kerja dari proses modifikasi AODV. Ketika protokol AODV melakukan mekanisme pencarian *node* untuk pengiriman ulang / *rebroadcast* paket RREQ akan langsung dikirim begitu saja. Sedangkan pada AODV yang dimodifikasi ini akan ada proses penyeleksian *node*. Seleksi *node* dilakukan setelah *node* sumber mem-*broadcast* paket RREQ ke semua *node* θngga dan *node* θngga telah menerima paket RREQ-nya. Dengan cara membagi *node* θngga yang sudah menerima paket RREQ-nya tadi menjadi 2 jenis, yaitu *reliable node* dan *unreliable node*. *Reliable node* adalah *node* yang berhak untuk mengirim ulang / *rebroadcast* RREQ yang diterimanya. Sedangkan *unreliable node* adalah *node* yang tidak berhak untuk mengirim ulang / *rebroadcast* RREQ yang diterimanya.

Pembagian ini dilakukan dengan cara membandingkan antara kecepatan milik *node* pengirim dengan kecepatan dari semua *node* penerima. Dimana kecepatan *node* penerima dan *node* pengirim akan membentuk sebuah sudut θ yang nantinya dibandingkan dengan *threshold* yang sudah ditentukan untuk mengetahui apakah *node* penerima dapat meneruskan paket RREQ yang sudah diterimanya atau tidak. Jika sudut θ yang terbentuk sudah sama atau kurang dari *threshold* yang sudah ditentukan maka *node* penerima termasuk dalam golongan *reliable node* sehingga dapat meneruskan paket RREQ yang diterimanya. Sedangkan jika hal yang terjadi malah kebalikannya maka *node* penerima termasuk dalam golongan *unreliable node* sehingga tidak dapat meneruskan paket RREQ yang sudah diterimanya.



Gambar 3.2 Alur Perancangan AODV Modifikasi

3.3.1 Perancangan Pengambilan Vektor Kecepatan

Pada Tugas Akhir ini pengambilan kecepatan dilakukan dengan cara membedakan antara semua *node* yang ada menjadi dua *node* utama, yaitu *node* pengirim dan *node* penerima. *Node* pengirim adalah *reliable node* yang sudah diberikan hak untuk melanjutkan paket RREQ yang diterimanya ke *node* lain agar paket tersebut dapat menuju ke *node* tujuan. Sedangkan *node* penerima adalah *node* yang terhubung dengan *node* pengirim namun belum

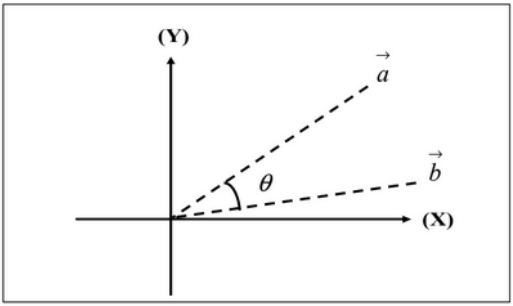
dipastikan apakah diperbolehkan untuk meneruskan paket RREQ atau tidak. Pada MANET semua *node* memiliki kebebasan arah gerak, maka kita harus mendapatkan kecepatan sesuai dengan arah gerak dari *node* tersebut. Arah gerak yang kita gunakan adalah arah gerak horizontal serta arah gerak vertikal. Hasil terakhir yang akan kita dapatkan adalah vektor kecepatan dari *node* pengirim dan *node* penerima. Pengambilan vektor kecepatan ini akan dikerjakan secara berurutan antara *node* pengirim dan semua *node* penerima yang memungkinkan untuk terhubung dengan *node* pengirim. Sampai *node* pengirim memiliki *node* penerima yang dapat membentuk sudut θ paling kecil atau sama dengan *threshold* yang sudah ditentukan.

```
node_pengirim = get_node(address)
if node_pengirim != 0
  load_velocity = node_pengirim(velo)
  while node_pengirim(index) != 0
      flag = node_pengirim(index)
      node_penerima = get_node(flag)
      load velocity2 = node_penerima(velo)
endif
```

Gambar 3.3 *Pseudocode* Pengambilan Kecepatan

3.3.2 Perancangan Nilai Sudut Teta (θ)

Pada Gambar 3.2 terdapat alur dalam melakukan penghitungan sudut θ. Sudut θ ini akan dibentuk oleh kecepatan *node* pengirim dan *node* penerima pada arah gerak vertikal (y) serta arah gerak horizontal (x) yang sudah diterima sebelumnya dan menjadikan titik arah gerak (0,0) pada peta seperti pada Gambar 3.4.



Gambar 3.4 Sudut Teta Pada Vektor Kecepatan a dan Vektor Kecepatan b

Dimana sudut θ sendiri akan merepresentasikan kemiripan kecepatan antara pengirim RREQ dan penerima RREQ. Semakin kecil sudut θ maka semakin mirip kecepatan antara keduanya dan semakin aman jika dibentuk sebuah rute untuk melaluinya. Sudut θ ini dapat dihitung dengan persamaan berikut :

$$\theta = \arccos\left(\frac{\vec{a}.\vec{b}}{|\vec{a}|.|\vec{b}|}\right) \quad (3.1)$$

Keterangan:
a = kecepatan pengirim / rebroadcaster RREQ
b = kecepatan penerima RREQ

3.3.3 Perancangan Pemilihan *Reliable Node* dan *Unreliable Node*

Setelah melakukan penghitungan sudut θ, akan dilakukan pemilihan apakah *node* tersebut termasuk *reliable node* atau *unreliable node.* Dimana *reliable node* adalah *node* yang berhak untuk melakukan *rebroadcast* paket RREQ. Sedangkan *unreliable node* adalah *node* yang tidak berhak untuk melakukan *rebroadcast* paket RREQ dikarenakan *node* ini bisa saja memiliki *link* yang *corrupt.* Pemilihan akan dilakukan dengan cara membandingkan sudut θ yang terbentuk oleh kecepatan *node* pengirim dan *node*

penerima dengan *threshold* atau batas nilai yang sudah ditentukan. Apabila sudut θ yang dibentuk kurang dari atau sama dengan *threshold* maka ia akan digolongkan *reliable node*, sedangkan jika sudut θ yang terbentuk lebih dari *threshold* maka ia digolongkan *unreliable node.*

```
if sudut_θ <= threshold
      continue proses
endif
elif sudut_θ > threshold
      drop RREQ packet
endelif
```

Gambar 3.5 *Pseudocode* Pemilihan *Reliable Node* dan *Unreliable Node*

3.4 Perancangan Simulasi pada NS-2

Simulasi MANETs pada NS-2 dilakukan dengan menggabungkan *file* skenario yang telah dibuat menggunakan NS-2 dan *file* skrip dengan ekstensi .tcl yang berisikan konfigurasi lingkungan simulasi.

Kode yang diubah diantaranya adalah pencarian jumlah *node* θngga pada *file* node.cc dan perbandingan dengan *threshold* pada *file* aodv.cc. Pada saat simulasi NS-2 dijalankan, maka *routing protocol* AODV akan menyeleksi *node* mana saja yang berhak melakukan *rebroadcast* paket RREQ.

3.5 Perancangan Metrik Analisis

Berikut ini merupakan parameter – parameter yang akan dianalisis pada Tugas Akhir ini untuk dapat membandingkan performa dari *routing protocol* AODV yang asli dengan AODV yang telah dimodifikasi:

3.5.1 *Packet Delivery Ratio* (PDR)

Packet delivery ratio merupakan perbandingan dari jumlah paket data yang dikirim dengan paket data yang diterima. PDR dapat menunjukkan keberhasilan paket yang dikirimkan. Semakin tinggi PDR artinya semakin berhasil pengiriman paket yang dilakukan Rumus untuk menghitung PDR dapat dilihat pada persamaan 3.1.

$$PDR = \frac{received}{sent} x\ 100\ \% \tag{3.2}$$

Keterangan:
PDR : *Packet Delivery Ratio*
received : banyak paket data yang diterima
sent : banyak paket data yang dikirimkan

3.5.2 *Average End-to-End Delay* (E2E)

Average End-to-End Delay dihitung berdasarkan rata-rata *delay* antara waktu paket data diterima dan waktu paket dikirimkan dalam satuan detik. Delay tiap paket didapat dari rentang waktu antara *node* asal saat mengirimkan paket dan *node* tujuan menerima paket. Delay tiap paket tersebut semua dijumlahkan dan dibagi dengan jumlah paket yang berhasil diterima, maka akan didapatkan rata – rata E2E, yang dapat dihitung dengan persamaan 3.2.

$$E2E = \frac{\sum_{m=1}^{recvnum} CBRRecvTime - CBRSentTime}{recvnum} \tag{3.3}$$

Keterangan:
E2E : *End-to-End Delay*
CBRRecvTime : Waktu *node* asal mengirimkan paket
CBRSentTime : Waktu *node* tujuan menerima paket
recvnum : Jumlah paket yang berhasil diterima

3.5.3 *Routing Overhead* (RO)

Routing Overhead adalah jumlah paket kontrol *routing* yang ditransmisikan per data paket ke *node* tujuan selama simulasi terjadi. *Routing Overhead* didapatkan dengan menjumlahkan semua paket kontrol *routing* yang ditransmisikan, baik itu paket *route request* (RREQ), *route reply* (RREP), maupun *route error* (RERR).. Perhitungan *Routing Overhead* dapat dilihat dengan persamaan 3.3.

$$RO = \sum_{m=1}^{sentnum} packet\ sent \tag{3.4}$$

Keterangan:
Packet sent : Jumlah kontrol paket RREQ, RREP, RERR
CBRRecvTime : Waktu *node* asal mengirimkan paket
CBRSentTime : Waktu *node* tujuan menerima paket
recvnum : Jumlah paket yang berhasil diterima

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi dari perancangan yang sudah dilakukan pada bab sebelumnya. Implementasi berupa kode sumber untuk membangun program.

4.1 Implementasi Skenario Mobilitas

Implementasi skenario mobilitas MANETs dibagi menjadi tiga, yaitu skenario *threshold* terbaik yang menggunakan AODV yang telah di modifikasi dengan ambang batas sudut θ sebagai pembanding untuk mendapatkan *threshold* terbaik untuk digunakan dus skenario setelahnya, skenario *node* yang menggunakan jumlah *node* sebagai pembanding dengan menggunakan *threshold* yang telah ditemukan pada percobaan petama dan skenario *speed* yang menggunakan kecepatan maksimum acak sebagai pembanding dengan menggunakan *threshold* yang telah ditemukan pada percobaan petama.

4.1.1 Skenario *Threshold* Terbaik

Dalam mengimplementasikan skenario *threshold* terbaik pada Tugas Akhir ini, penulis membuat skenario dengan luas area 1000 m x 1000 m dengan pengaturan jumlah *node* sebesar 100 serta memerlukan pengaturan kecepatan maksimum acak 20 m/s perskenario dengan bantuan NS-2. Dapat dilihat pada Gambar 4.3.

```
./setdest -v 2 -n 18 -s 1 -m 5 -M 100 -t 200 -P
1 -p 1 -x 1000 -y 1000 > setdest100.tcl
```

Gambar 4.1 Perintah Setdest Skenario *Threshold* Terbaik

Dengan rincian pengaturan sebagai berikut :
-v: versi = 2

-n: jumlah *node* = 1000 *node*
-s: *speed type* = 1 (*uniform*)
-m: *min speed* = 5 m/s
-M: *max speed* = 20 m/s
-t: *simulation time* = 200 detik
-P: *pause type* = 1 (constant)
-p: *pause time* = 0
-x: maksimum X = 1000 meter
-y: maksimum Y = 1000 meter

Pada skenario ini juga kita harus memodifikasi AODV yang sudah kita modifikasi dengan mengatur rentang *threshold* yang diperlukan AODV modifikasi pada 0º, 33º hingga 165º. Untuk potongan kode tersebut dapat dilihat pada Gambar 4.4.

```
1.    //mengecek threshold
2.    double tetathreshold = 0;
3.    if(teta > tetathreshold){
4.    Packet::free(p); //drop packet
5.    Return;
6.    }
```

Gambar 4.2 Potongan Kode Pengaturan Besaran *Threshold*

4.1.2 Skenario *Node*

Dalam mengimplementasikan skenario *node* pada Tugas Akhir ini, penulis membuat skenario dengan luas area 1000 m x 1000 m dengan pengaturan kecepatan maksimum acak sebesar 20 m/s serta memerlukan pengaturan jumlah *node* yang terus bertambah dengan rentang 20, 40 hingga 200 *node* perskenario dengan bantuan NS-2. Dapat dilihat pada Gambar 4.1.

```
./setdest -v 2 -n 18 -s 1 -m 5 -M 20 -t 200 -P
1 -p 1 -x 1000 -y 1000 > setdest20.tcl
```

Gambar 4.3 Perintah Setdest Skenario *Node*

Dengan rincian pengaturan sebagai berikut :
-v: versi = 2
-n: jumlah *node* = 20 sampai 200 *node*
-s: *speed type* = 1 (*uniform*)
-m: *min speed* = 5 m/s
-M: *max speed* = 20 m/s
-t: *simulation time* = 200 detik
-P: *pause type* = 1 (*constant*)
-p: *pause time* = 0
-x: maksimum X = 1000 meter
-y: maksimum Y = 1000 meter

4.1.3 Skenario *Speed*

Dalam mengimplementasikan skenario *speed* pada Tugas Akhir ini, penulis membuat skenario dengan luas area 1000 m x 1000 m dengan pengaturan jumlah *node* sebesar 100 serta memerlukan pengaturan kecepatan maksimum acak yang terus bertambah dengan rentang 5, 20, 40 hingga 100 m/s perskenario dengan bantuan NS-2. Dapat dilihat pada Gambar 4.2.

```
./setdest -v 2 -n 18 -s 1 -m 5 -M 100 -t 200 -P
1 -p 1 -x 1000 -y 1000 > setdest100.tcl
```

Gambar 4.4 Perintah Setdest Skenario *Speed*

Dengan rincian pengaturan sebagai berikut :
-v: versi = 2
-n: jumlah *node* = 1000 *node*
-s: *speed type* = 1 (*uniform*)
-m: *min speed* = 5 m/s
-M: *max speed* = 20 m/s
-t: *simulation time* = 200 detik
-P: *pause type* = 1 (constant)
-p: *pause time* = 0

-x: maksimum X = 1000 meter
-y: maksimum Y = 1000 meter

4.2 Implementasi Modifikasi pada *Routing Protocol* AODV untuk Menentukan *Reliable Node*

Pada Tugas Akhir ini dilakukan modifikasi pada *routing protocol* AODV agar dapat mengurangi jumlah *unreliable node*, yaitu *node* yang kemungkinan berisi *link* yang rusak sehingga dapat membuat *routing overhead* pada AODV meningkat. Supaya *node* tersebut tidak dapat melakukan *rebroadcast* paket RREQ. Hal tersebut dilakukan dengan cara memilih *reliable node* berdasarkan *velocity-aware* dengan cara membandingkan kecepatan antara *node* pengirim dengan *node* penerima hingga membentuk sudut θ yang nantinya dibandingkan dengan *threshold* yang sudah ditentukan, sehingga dapat dilihat peningkatan performa pada *routing* AODV yang telah dimodifikasi.

Implementasi modifikasi *routing protocol* AODV ini dibagi menjadi 3 bagian yaitu:

- Implementasi Pengambilan Kecepatan *Node* Pengirim dan *Node* Penerima
- Implementasi Penghitungan Sudut θ
- Implementasi Pemilihan *Reliable Node* dan *Unreliable Node*

Kode implementasi dari *routing protocol* AODV pada NS-2 versi 2.35 berada pada direktori ns-2.35/aodv. Pada direktori tersebut terdapat beberapa file diantaranya seperti aodv.cc, aodv.h dan sebagainya. Pada Tugas Akhir ini, penulis memodifikasi *file* aodv.cc yang terdapat dalam *folder* ns-2.35/aodv untuk mengambil kecepatan *node* pengirim dan *node* penerima, menghitung sudut θ, memilih *reliable node* dan *unreliable node* dan *file* aodv.h yang ada di dalam *folder* ns-2.35/aodv untuk mendaftarkan fungsi dan *timer* baru. Pada bagian ini penulis akan menjelaskan langkah – langkah dalam mengimplementasikan modifikasi *routing protocol*

AODV untuk mengurangi jumlah *forwarding node* yang melakukan *rebroadcast route request*.

4.2.1 Implementasi Pengambilan Vektor Kecepatan *Node* Pengirim dan *Node* Penerima

Ketika *node* sumber ingin mengirimkan paket RREQ, *node* tersebut harus mengirimkan paket RREQ terlebih dahulu ke *node* penerima yang berada di dekatnya. Pada saat itulah *node* dibedakan menjadi 2, yaitu *node* pengirim dan *node* penerima.

Selanjutnya kita harus menentukan arah yang dituju oleh kedua node tersebut dengan cara mengambil kecepatan pada arah gerak horizontal dan arah gerak vertikal yang dimiliki oleh *node* pengirim dan *node* penerima. Pada penerapannya seringkali *node* hanya bergerak terhadap satu arah gerak saja sehinga menyebabkan kecepatan pada arah gerak lainnya tidak ada. Apabila suatu arah gerak tidak memiliki kecepatan maka sudut θ tidak dapat terbentuk. Sehingga kita melakukan modifikasi dengan mengganti kecepatan yang ada apabila *node* tersebut tidak menghasilkan kecepatan pada satu arah gerak tertentu. Pengambilan vektor kecepatan dilakukan di *file* aodv.cc yang terdapat dalam folder ns2.3.5/aodv. Untuk potongan kode tersebut bisa dilihat pada Gambar 4.5. Kode pada *file* aodv.cc dapat dilihat di lampiran A.1.

```
1. //mendapatkan kecepatan pengirim RREQ
2. double sendervx, sendervy, sendervz;
3. Node* sender =
   Node::get_node_by_address(ih->saddr());
4. ((MobileNode *)sender)-
   >getVelo(&sendervx, &sendervy,
   &sendervz);
5.
6. //mendapatkan kecepatan penerima RREQ
7. double receivervx, receivervy,
   receivvervz
```

```
8. Node* receiver =
   Node::get_node_by_address(index)
9. ((MobileNode *)receiver)-
   >getVelo(&receivervx, &receivervy,
   &receivervz)
10.   //agar tidak error ketika dibagi 0
11.   If(sendervx == 0) sendervx = 0.0001;
12.   If(sendervy == 0) sendervy = 0.0001;
13.   If(receivervx == 0) receivervx =
   0.0001;
14.   If(receivervy == 0) receivervy =
   0.0001;
```

Gambar 4.5 Potongan Kode Pengambilan *Node* Pengirm dan *Node* Penerima

4.2.2 Implementasi Penghitungan Sudut Teta

Pada tahap selanjutnya setelah mengetahui vektor kecepatan *node* pengirim dan *node* penerima pada arah gerak horizontal dan arah gerak vertikal serta telah memastikan bahwa kecepatan tersebut terhindar oleh angka kosong. Maka langkah yang harus dilakukan adalah menghitung sudut θ yang dibentuk oleh kedua kecepatan tersebut dengan arah gerak (0,0) sebagai pusat sudut. Penghitungan ini dilakukan pada fungsi yang terletak pada kode sumber aodv.cc di ns2.35/aodv. Potongan kode untuk proses perhitungan sudut θ dapat dilihat pada Gambar 4.6.

```
1.    //rumus sudut teta
2.    double PI = 3.14159265;
3.    double pembilang = sendervx * receivervx +
      sendervy * receivervy;
4.    double penyebut =
      sqrt(pow(sendervx,2)+pow(sendervy,2))*sqrt
      (pow(receivervx,2)+pow(receivervy,2));
5.    double teta = acos(pembilang/penyebut);
```

Gambar 4.6 Potongan Kode Perhitungan Sudut *Teta*

4.2.3 Implementasi Penghitungan Pemilihan *Reliable Node* dan *Unreliable Node*

Pada tahap selanjutnya setelah dapat mengetahui sudut θnya, langkah yang harus dilakukan selanjutnya adalah pemilihan *reliable node* dan *unreliable node* melalui perantara *threshold* yang sudah ditentukan terlebih dahulu untuk mengetahui *node* mana yang berhak melanjutkan paket RREQ.

Penghitungan ini dilakukan pada kode sumber aodv.cc yang terletak pada ns2.35/aodv. Apabila sudut θ kurang dari atau sama dengan *threshold* yang diberikan, maka *node* tersebut menjadi *reliable node* dan sebaliknya jika sudut θ lebih dari *threshold* maka termasuk *unreliable node*. *Reliable node* berhak *rebroadcast* paket RREQ yang sudah diterimanya dan *unreliable node* harus ditekan penggunaannya, Potongan kode untuk proses seleksi *reliable node* dapat dilihat pada Gambar 4.7.

```
1.    //mengecek threshold
2.    double θthreshold = 165;
3.    if(θ > θthreshold){
4.       Packet::free(p); //drop packet
5.       Return;
6.    }
```

Gambar 4.7 Potongan Kode Penyeleksian *Reliable* dan *Unreliable Node*

4.3 Implementasi Simulasi pada NS-2

Implementasi simulasi MANETs diawali dengan pendeskripsian lingkungan simulasi pada sebuah *file* tcl. *File* ini berisikan konfigurasi setiap *node* dan langkah-langkah yang dilakukan selama simulasi. Potongan konfigurasi lingkungan simulasi dapat dilihat pada Gambar 4.8.

```
set val(chan)    Channel/WirelessChannel
set val(prop)    Propagation/TwoRayGround
set val(netif)   Phy/WirelessPhy
set val(mac)     Mac/802_11
set val(ifq)     Queue/DropTail/PriQueue
set val(ll)      LL
set val(ant)     Antenna/OmniAntenna
set opt(x)       1000
set opt(y)       1000
set val(ifqlen)  50
set val(nn)      20
set val(seed)    1.0
set val(adhocRouting) AODV
set val(stop)    200
set val(cp)      "cbr.tcl"
set val(sc)      "setdest20.tcl"
```

Gambar 4.8 Implementasi Simulasi NS-2

Pada konfigurasi dilakukan pemanggilan terhadap *file traffic* yang berisikan konfigurasi *node* asal, *node* tujuan, pengiriman paket, serta *file* skenario yang berisi pergerakan *node* yang telah di*generate*. Kode implementasi pada NS-2 dapat dilihat pada lampiran A.5 Kode Skenario NS-2.

Konfigurasi untuk *file traffic* bisa dilakukan dengan membuat *file* berekstensi .txt untuk menyimpan konfigurasi tersebut. Pada *file* konfigurasi lingkungan simulasi, *file traffic*

tersebut dimasukkan agar dibaca sebagai *file traffic*. Potongan konfigurasi *file traffic* dapat dilihat pada Gambar 4.9.

```
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(18) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(19) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 0.20000000000000001
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 2.5568388786897245 "$cbr_(0) start"
```

Gambar 4.9 Implementasi Simulasi File Traffic

Pada konfigurasi tersebut, ditentukan *node* sumber dan *node* tujuan pengiriman paket. Pengiriman dimulai pada detik ke-2.55. Implementasi konfigurasi *file traffic* untuk simulasi pada NS-2 dapat dilihat pada lampiran A.3 Kode Konfigurasi *Traffic*

4.4 Implementasi Metrik Analisis

Simulasi yang telah dijalankan oleh NS-2 menghasilkan sebuah *trace file* yang berisikan data mengenai apa saja yang terjadi selama simulasi dalam bentuk *file* berekstensi .tr. Dari data *trace file* tersebut, dapat dilakukan analisis performa *routing protocol* dengan mengukur beberapa metrik. Pada Tugas Akhir ini, metrik yang akan dianalisis adalah *Packet Delivery Ratio, End to End Delay,* dan *Routing Overhead*.

4.4.1 Implementasi *Packet Delivery Ratio* (PDR)

Pada subbab 2.3.2 telah ditunjukkan contoh struktur data *event* yang dicatat dalam *trace file* oleh NS-2. Kemudian, pada persamaan 3.1 telah dijelaskan bagaimana menghitung PDR. Skrip awk untuk menghitung PDR berdasarkan kedua informasi tersebut dapat dilihat pada lampiran A.4 Kode Skrip AWK *Packet Delivery Ratio*.

PDR didapatkan dengan cara menghitung setiap baris terjadinya *event* pengiriman dan penerimaan paket data yang dikirim melalui agen pada *trace file*. Skrip menyaring setiap baris yang mengandung *string* AGT karena kata kunci tersebut menunjukan *event* yang berhubungan dengan paket komunikasi data. Penghitungan dilakukan dengan menjumlahkan paket yang dikirimkan dan paket yang diterima dengan menggunakan karakter pada kolom pertama sebagai *filter*. Kolom pertama menunjukan event yang terjadi dari sebuah paket. Setelah itu nilai PDR dihitung dengan cara persamaan 3.1. Pseudocode untuk menghitung PDR dapat dilihat pada Gambar 4.10.

Contoh perintah pengeksekusian skrip awk untuk menganalisis *trace file* adalah `awk -f pdr.awk result.tr`.

```
sent = 0
received = 0
for i = 1 to the number of rows
     if in a row contains “s” and AGT then
          sent++
     else if in a row contains “r” and AGT then
          received++
     end if
pdr = received / sent
```

Gambar 4.10 Pseudocode untuk Perhitungan PDR

4.4.2 Implementasi *Average End-to-End Delay (E2E)*

Skrip awk untuk menghitung E2E dapat dilihat pada lampiran A.5 Kode Skrip AWK Rata-Rata *End-to-End Delay*.

Dalam perhitungan E2E, langkah yang digunakan untuk mendapatkan E2E hampir sama dengan ketika mencari PDR, hanya saja yang perlu diperhatikan adalah waktu dari sebuah *event* yang tercatat pada kolom ke-2 dengan *filter event* pada kolom ke-4 adalah layer AGT dan *event* pada kolom pertama guna membedakan paket dikirim atau diterima. Setelah seluruh baris yang memenuhi didapatkan, akan dihitung *delay* dari paket dengan mengurangi waktu dari paket diterima dengan waktu dari paket dikirim dengan syarat memiliki *id* paket yang sama.

Setelah mendapatkan *delay* paket, langkah selanjutnya adalah dengan mencari rata-rata dari *delay* tersebut dengan menjumlahkan semua *delay* paket dan membaginya dengan jumlah paket. *Pseudocode* untuk menghitung rata-rata E2E dapat dilihat pada Gambar 4.11.

Contoh perintah pengeksekusian skrip awk untuk menganalisis *trace file* adalah `awk –f e2e.awk result.tr`.

```
sum_delay = 0
counter = 0

for i = 1 to the number of rows
     counter++
     if layer == AGT and event == s then
      start_time[packet_id] = time
     else if layer == AGT and event == r then
      end_time[packet_id] = time
     end if
     delay[packet_id] = end_time[packet_id] –
start_time[packet_id]
     sum_delay += delay[packet_id]
e2e = sum_delay / counter
```

Gambar 4.11 Pseudocode untuk Perhitungan E2E

4.4.3 Implementasi *Routing Overhead* (RO)

Seperti yang telah dijelaskan sebelumnya, *routing overhead* merupakan jumlah dari paket kontrol *routing* baik itu RREQ, RREP, maupun RERR. Dengan begitu, untuk mendapatkan RO yang perlu dilakukan adalah menjumlahkan tiap paket dengan *filter event sent* pada kolom pertama dan *event layer* RTR pada kolom ke-4. Perhitungan RO telah dijelaskan pada persamaan 3.3. Skrip AWK untuk menghitung RO dapat dilihat pada lampiran A.6 Kode Skrip AWK *Routing Overhead*. *Pseudocode* untuk menghitung RO dapat dilihat pada Gambar 4.12.

Contoh perintah pengeksekusian skrip awk untuk menganalisis *trace file* adalah `awk -f ro.awk result.tr`.

```
ro = 0
for i = 1 to the number of rows
     if in a row contains "s" and RTR then
        ro++
     end if
```

Gambar 4.12 Pseudocode Perhitungan Routing Overhead

BAB V
UJICOBA DAN EVALUASI

Pada bab ini akan dilakukan tahap ujicoba dan evaluasi sesuai dengan rancangan dan implementasi. Dari hasil yang didapatkan setelah melakukan uji coba, akan dilakukan evaluasi sehingga dapat ditarik kesimpulan pada bab selanjutnya.

5.1 Lingkungan Uji Coba

Uji coba dilakukan pada perangkat dengan spesifikasi seperti yang tertera pada Tabel 5.1.

Tabel 5.1 Spesifikasi Perangkat yang Digunakan

Komponen	**Spesifikasi**
CPU	Intel(R) Core™ i7-7500U CPU @ 2.70GHz
Sistem Operasi	Ubuntu 18.04.2 LTS
Linux Kernel	Linux kernel 4.4
Memori	8.0 GB
Penyimpanan	123 GB

Adapun versi perangkat lunak yang digunakan dalam Tugas Akhir ini adalah sebagai berikut:

- NS-2 versi 2.35 untuk simulasi skenario MANETs.

Parameter lingkungan uji coba yang digunakan pada NS-2 dapat diihat pada Tabel 5.2. Pengujian dilakukan dengan menjalankan skenario yang disimulasikan pada NS-2. Dari simulasi tersebut dihasilkan sebuah *trace file* dengan ekstensi .tr yang akan dianalisis dengan bantuan skrip awk untuk mendapatkan PDR, E2E, dan RO menggunakan kode yang terdapat pada lampiran A.4 Kode Skrip AWK *Packet Delivery Ratio*, A.5 Kode Skrip AWK Rata-Rata *End-to-End Delay*, dan A.6 Kode Skrip AWK *Routing Overhead*.

Tabel 5.2 Lingkungan Uji Coba

No.	Parameter	Spesifikasi
1	Network simulator	NS-2.35
2	*Routing protocol*	AODV
3	Waktu simulasi	200 detik
4	Area simulasi	1000 m x 1000 m
5	Jumlah *Node*	20, 60, 80 100, 120, 140, 160, 180, 200 (Skenario *node*) 100 (Skenario *speed*)
6	Kecepatan minimum	5 m/s
7	Kecepatan maksimum acak	20 m/s (Skenario *node*) 5 m/s, 20 m/s, 40 m/s, 60 m/s, 80 m/s, 100 m/s (Skenario *speed*)
8	*Threshold*	165º (Skenario *node*, Skenario *speed*) 0º, 33º, 66º, 99º, 132º, 165º (Uji Coba *Threshold*)
9	Paket Interval	1 paket/detik
10	Protokol MAC	IEEE 802.11p
11	Model Propagasi	*Two-ray ground*

5.2 Hasil Uji Coba

Hasil uji coba menggunakan *node* sumber dan *node* tujuan yang diletakkan secara statis. Hasil dari skenario *node* dan skenario *speed* untuk Tugas Akhir ini dapat dilihat sebagai berikut:

5.2.1 Hasil Uji Coba Skenario *Threshold* Terbaik

Pengujian pada skenario *threshold* terbaik digunakan untuk melihat perbandingan PDR, E2E dan RO antara *routing protocol*

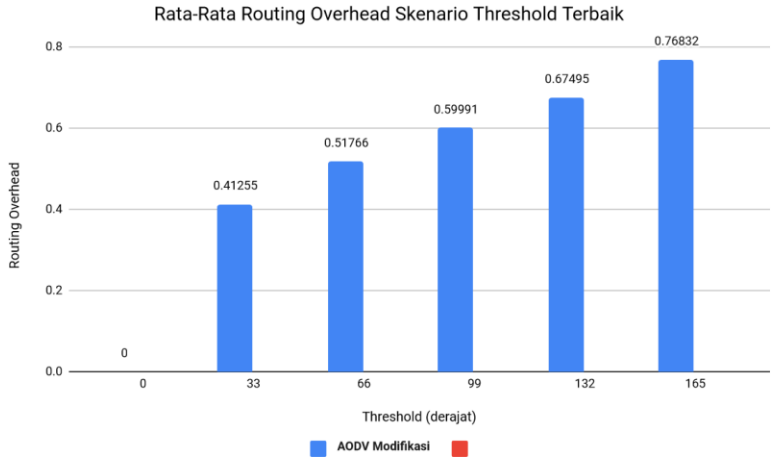
AODV asli dan AODV yang telah dimodifikasi dalam pemilihan *node* yang dapat menerima paket *route request*.

Pengambilan data uji PDR, E2E dan RO pada skenario *real* dilakukan sebanyak 10 kali dengan skenario mobilitas *random* pada skenario *speed* dengan luas area 1000 m x 1000 m dengan *node* sebanyak 100 dan kecepatan maksimum acak 20 m/s. Untuk hasil analisis skenario *threshold* terbaik dengan sudut 0º, 33º, 66º, 99º, 132º dan 165º dapat dilihat pada Tabel 5.3, Tabel 5.4 dan Tabel 5.5.

Tabel 5.3 Hasil Rata - Rata *Packet Delivery Ratio* Skenario *Threshold* Terbaik

Threshold **(derajat)**	**AODV Modifikasi**
0	0
33	0.41255
66	0.51766
99	0.59991
132	0.67495
165	0.76832

Dari data pada Tabel 5.3, dibuat grafik yang merepresentasikan hasil perhitungan PDR yang ditunjukkan pada Gambar 5.1.



Gambar 5.1 Grafik *Packet Delivery Ratio* Skenario *Threshold* Terbaik

Berdasarkan grafik pada Gambar 5.1 dapat dilihat bahwa *routing protocol* AODV yang telah dimodifikasi mengalami kenaikan yang signifikan pada *packet delivery ratio*. Pada lingkungan *threshold* dengan sudut 0º, 33º, 66º, 99º, 132º dan 165º berturut-turut 0, 0.41255, 0.51766, 0.59991, 0.67495 dan 0.76832 dengan kenaikan tiap bertambahnya sudut masing-masing 25.48%, 15.89%, 12.51% dan 13.83%.

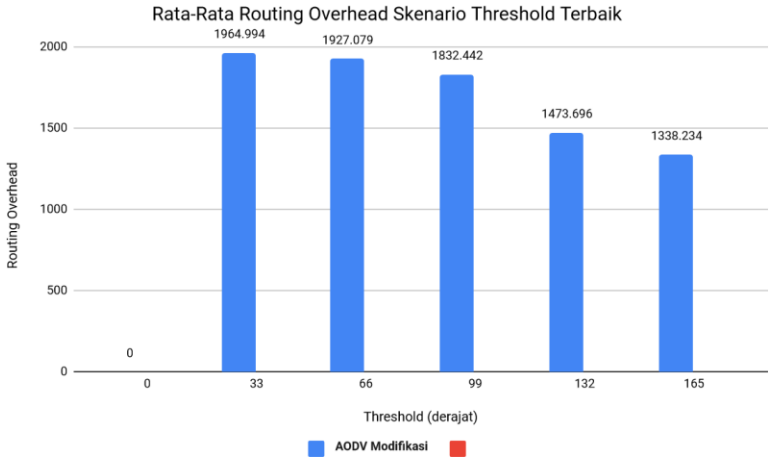
Dapat dilihat rata-rata kenaikan yang terjadi untuk PDR adalah 16.93%. Dapat dilihat juga pada AODV modifikasi hasil PDR tergolong bertambah dengan stabil, meski dapat kita menarik kesimpulan dengan menggunakan *threshold* pada sudut 165º hasil PDR akan semakin bagus. Hal itu disebabkan banyak *node* yang terbuang percuma pada *threshold* dibawah sudut 165º. Dimana ada kemungkinan *reliable node* yang memungkinkan juga dibuang, hingga pada *threshold* 0º hampir seluruh *node* dibuang hingga tidak ada *node* lagi yang tersisa untuk mengirimkan packet menuju *node* destinasi. Dengan banyaknya *node* yang dibuang juga ditakutkan rute pencarian akan semakin berlangsung lama dan tidak menutup kemungkinan paket tidak dapat tersampaikan.

Hasil pengambilan data rata-rata untuk *end-to-end delay* (E2E) pada skenario *threshold* terbaik dengan sudut 0º, 33º, 66º, 99º, 132º dan 165º dapat dilihat pada Tabel 5.4.

Tabel 5.4 Hasil Rata -Rata End-To-End Delay Skenario *Threshold* Terbaik

Threshold **(derajat)**	**AODV Modifikasi**
0	0
33	1964.994
66	1927.079
99	1832.442
132	1473.696
165	1338.234

Dari data pada Tabel 5.4, dibuat grafik yang merepresentasikan hasil perhitungan *end-to-end delay* yang ditunjukkan pada Gambar 5.2.



Gambar 5.2 Grafik *End-to-end Delay* pada Skenario *Threshold* Terbaik

Berdasarkan grafik pada Gambar 5.2 dapat dilihat bahwa *routing protocol* AODV yang telah dimodifikasi mengalami penurunan yang signifikan pada *end-to-end delay*. Pada lingkungan *threshold* dengan sudut 0º, 33º, 66º, 99º, 132º dan 165º berturut-turut 0, 1964.994, 1927.079, 1832.442, 1473.696 dan 1338.234 dengan penurunan tiap bertambahnya sudut masing-masing 1.93%, 4.91%, 19.58% dan 9.19%.

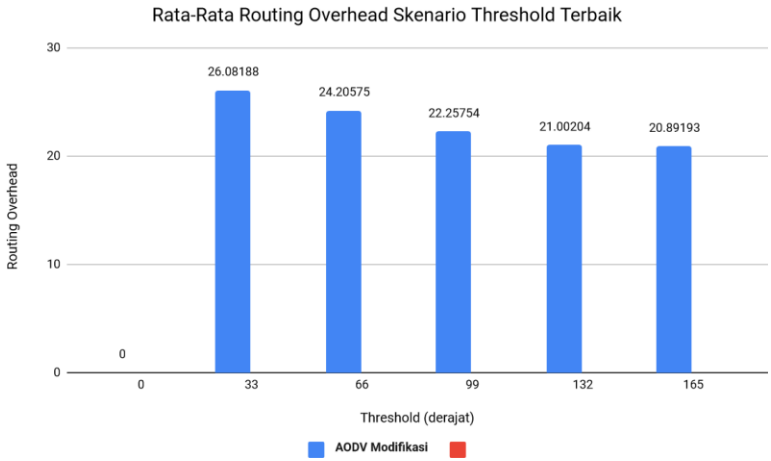
Dapat dilihat rata-rata penurunan yang terjadi untuk E2E adalah 8.90%. Dapat dilihat juga pada AODV modifikasi hasil E2E tergolong berkurang dengan stabil. Meskipun *threshold* pada sudut 132º dan sudut 165º pengurangan yang terjadi sangat sedikit, kita masih dapat menarik kesimpulan dengan menggunakan *threshold* pada sudut 165º hasil E2E akan semakin bagus. Hal itu disebabkan banyak *node* yang terbuang percuma pada *threshold* dibawah sudut 165º. Dimana ada kemungkinan *reliable node* yang memungkinkan juga dibuang, hingga pada *threshold* 0º hampir seluruh *node* dibuang hingga tidak ada *node* lagi yang tersisa untuk mengirimkan packet menuju *node* destinasi. Dengan banyaknya *node* yang dibuang juga ditakutkan rute pencarian akan semakin berlangsung lama dan tidak menutup kemungkinan paket tidak dapat tersampaikan.

Untuk hasil pengambilan data *routing overhead* (RO) pada skenario *threshold* terbaik dengan sudut 0º, 33º, 66º, 99º, 132º dan 165º dapat dilihat pada Tabel 5.5.

Tabel 5.5 Hasil Rata - Rata *Routing Overhead* Skenario *Threshold* Terbaik

Threshold **(derajat)**	**AODV Modifikasi**
0	0
33	26.08188
66	24.20575
99	22.25754
132	21.00204
165	20.89193

Dari data pada Tabel 5.5, dibuat grafik yang merepresentasikan hasil perhitungan *routing overhead* yang ditunjukkan pada Gambar 5.3.



Gambar 5.3 Grafik *Routing Overhead* Skenario *Threshod* Terbaik

Berdasarkan grafik pada Gambar 5.3 dapat dilihat bahwa *routing protocol* AODV yang telah dimodifikasi mengalami penurunan yang signifikan pada *rioting overhead.* Pada lingkungan *threshold* dengan sudut 0º, 33º, 66º, 99º, 132º dan 165º berturut-turut 0, 26.08188, 24.20575, 22.25754, 21.00204 dan 20.89193 dengan penurunan tiap bertambahnya sudut masing-masing 7.19%, 8.05%, 5.64% dan 0.52%.

Dapat dilihat rata-rata penurunan yang terjadi untuk RO adalah 5.35%. Dapat dilihat juga pada AODV modifikasi hasil RO tergolong berkurang dengan stabil. Meskipun *threshold* pada sudut 132º dan sudut 165º pengurangan yang terjadi sangat sedikit, kita masih dapat menarik kesimpulan dengan menggunakan *threshold* pada sudut 165º hasil RO akan semakin bagus. Hal itu disebabkan banyak *node* yang terbuang percuma pada *threshold* dibawah sudut 165º. Dimana ada kemungkinan *reliable node* yang memungkinkan juga dibuang, hingga pada *threshold* 0º hampir seluruh *node* dibuang

hingga tidak ada *node* lagi yang tersisa untuk mengirimkan packet menuju *node* destinasi. Dengan banyaknya *node* yang dibuang juga ditakutkan rute pencarian akan semakin berlangsung lama dan tidak menutup kemungkinan paket tidak dapat tersampaikan dengan baik. Serta akan banyak paket RERR yang dikirm percuma hingga membanjiri jaringan.

5.2.2 Hasil Uji Coba Skenario *Node*

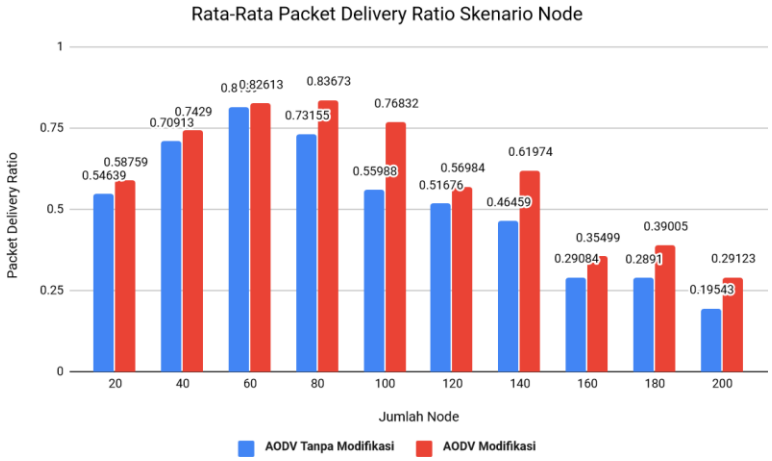
Pengujian pada skenario *node* digunakan untuk melihat perbandingan PDR, E2E, dan RO antara *routing protocol* AODV asli dan AODV yang telah dimodifikasi dalam pemilihan *node* yang dapat menerima paket *route request*.

Pengambilan data uji pada skenario *grid* dilakukan sebanyak 10 kali dengan skenario mobilitas *random* pada slenario *node* dengan luas area 1000 m x 1000m m dan *node* sebanyak 20, 40, 60, dan 80 untuk lingkungan jarang, 100, 120 dan 140 *node* untuk lingkungan yang sedang, serta 160, 180 dan 200 *node* untuk lingkungan padat dilakukan pada kecepatan standar yaitu 20 m/s. Hasil analisis dapat dilihat pada Tabel 5.6, Tabel 5.7, Tabel 5.8.

Tabel 5.6 Hasil Rata - Rata *Packet Delivery Ratio* Skenario *Node*

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**	**Perbedaan (AODV Modifikasi – AODV Asli)**
20	0.54639	0.58759	+ 0.0412
40	0.70913	0.7429	+ 0.0337
60	0.8159	0.82613	+ 0.0102
80	0.73155	0.83673	+ 0.1051
100	0.55988	0.76832	+ 0.2084
120	0.51676	0.56984	+ 0.0530
140	0.46459	0.61974	+ 0.1551
160	0.29084	0.35499	+ 0.0641
180	0.2891	0.39005	+ 0.1009
200	0.19543	0.29123	+ 0.0958

Dari data pada Tabel 5.6, dibuat grafik yang merepresentasikan hasil perhitungan PDR yang ditunjukkan pada Gambar 5.4.



Gambar 5.4 Grafik Packet Delivery Ratio Skenario *Node*

Berdasarkan grafik pada Gambar 5.4 dapat dilihat bahwa *routing protocol* AODV yang telah dimodifikasi dan juga *routing protocol* AODV asli mengalami kenaikan saat lingkungan jumlah *node* 20 hingga ke jumlah *node* 100 dan penurunan pada lingkungan *node* 120 hingga menyentuh jumlah *node* 200 packet delivery ratio. Pada lingkungan yang jarang dengan jumlah *node* 20, 40, 60 dan 80 menghasilkan perbedaan selisih PDR masing-masing sebesar 0.0412, 0.03377, 0.01023 dan 0.10518 atau naik masing-masing menjadi sekitar 7.5%, 4.75%, 0.02% dan 14.36% dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam hal PDR dari AODV asli. Pada lingkungan yang sedang dengan jumlah 100, 120 dan 140 node menghasilkan perbedaan selisih PDR masing-masing sebesar 0.2084, 0.0530 dan 0.1551 atau naik masing-masing menjadi sekitar 37.22%, 10.25% dan 33.38% dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam hal PDR dari AODV asli. Pada

lingkungan yang padat dengan jumlah 160, 180 dan 200 node menghasilkan perbedaan selisih PDR masing-masing sebesar 0.0641, 0.10095 dan 0.098 atau naik masing-masing menjadi sekitar 21.34%, 34.9% dan 49.02% dimana *routing protocol* AODV yang telah dimodifikasi juga unggul dalam hal PDR dari AODV asli.

Dapat dilihat rata-rata kenaikan yang terjadi untuk PDR adalah 21.34%. Jika ketiga lingkungan tersebut dibandingkan dapat dilihat bahwa dengan jumlah *node* yang lebih sedikit atau pada lingkungan dengan *node* yang jarang akan menghasilkan PDR yang lebih bagus daripada di lingkungan dengan jumlah *node* yang sedang maupun yang padat, baik untuk *routing protocol* AODV asli maupun *routing protocol* AODV yang telah dimodifikasi. Dengan ini kita dapat menyimpulkan PDR sangat terpengaruh dengan kepadatan *node* pada peta uji coba. Semakin padat *node* pada peta uji coba maka PDR yang dihasilkan semakin kecil. Adnya kemungkinan saling bertabrakan antar rute dan *node* lah yang menjadi penyebab utama hal ini terjadi. Kepadatan *node* selain berpengaruh pada AODV modifikasi juga berpengaruh pada AODV tanpa modifikasi. Dapat dilihat pula bahwa AODV yang telah dimodifikasi menghasilkan PDR yang lebih bagus daripada AODV asli dengan jumlah selisih PDR yang cukup signifikan.

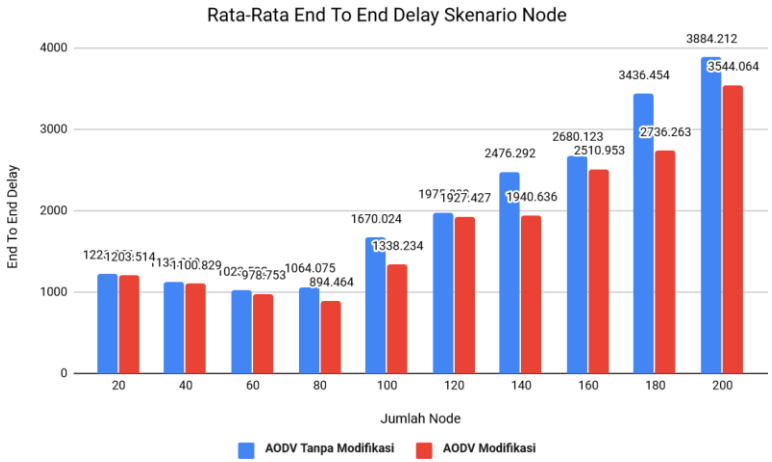
Hasil pengambilan data rata-rata untuk *end-to-end delay* (E2E) pada skenario *node* dengan jumlah *node* 20, 60, 80, 100, 120, 140, 160, 180, dan 200 dapat dilihat pada Gambar 5.5.

Tabel 5.7 Hasil Rata - Rata End-To-End Delay Skenario *Node*

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**	**Perbedaan (AODV Modifikasi – AODV Asli)**
20	1228.056	1203.514	-24.542
40	1133.069	1100.829	-32.24
60	1023.522	978.753	-44.769
80	1064.075	894.464	-169.611
100	1670.024	1338.234	-331.79
120	1975.003	1927.427	-47.576

140	2476.292	1940.636	-535.656
160	2680.123	2510.953	-169.17
180	3436.454	2736.263	-700.191
200	3884.212	3544.064	-340.148

Dari data pada Tabel 5.7, dibuat grafik yang merepresentasikan hasil perhitungan *end-to-end delay* yang ditunjukkan pada Gambar 5.5.



Gambar 5.5 Grafik End-to-end Delay Skenario *Node*

Berdasarkan grafik pada Gambar 5.5 dapat dilihat bahwa rata-rata *end-to-end delay* antara *routing protocol* AODV asli dan AODV yang telah dimodifikasi mengalami penurunan yang signifikan. Pada lingkungan yang jarang dengan jumlah *node* 20, *node* 40, *node* 60 dan *node* 80 menghasilkan perbedaan selisih *end-to-end delay* masing-masing sebesar 24.542, 32.24, 44.769 dan 169.611 atau turun masing-masing menjadi sekitar 2.00%, 2.85%, 4.37% dan 15.94% dimana *routing protocol* AODV yang telah dimodifikasi mengalami penurunan dalam hal *end-to-end delay* dari AODV asli.

Pada lingkungan yang sedang dengan jumlah 100, 120 dan 140 node menghasilkan perbedaan selisih *end-to-end delay* masing-masing sebesar 331.79, 47.576 dan 535.656 atau turun masing-masing menjadi sekitar 19.87%, 2.41% dan 21.63% dimana *routing protocol* AODV yang telah dimodifikasi mengalami penurunan dalam hal *end-to-end delay* dari AODV asli. Pada lingkungan yang padat dengan jumlah 160, 180 dan 200 node menghasilkan perbedaan selisih *end-to-end delay* masing-masing sebesar 169.17, 700.19 dan 340.148 atau turun masing-masing menjadi sekitar 6.31%, 20.38% dan 8.76% dimana *routing protocol* AODV yang telah dimodifikasi juga mengalami penurunan dalam hal *end-to-end delay* dari AODV asli.

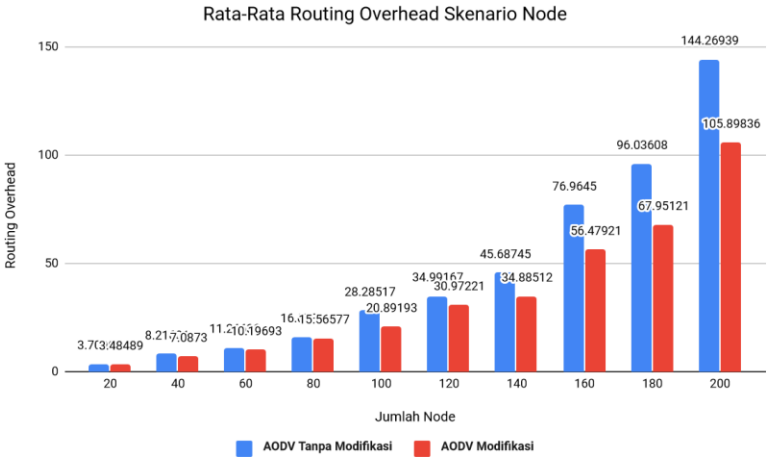
Dapat dilihat rata-rata penurunan yang terjadi untuk *end-to-end delay* adalah 10.45%. Jika ketiga lingkungan tersebut dibandingkan dapat dilihat bahwa dengan jumlah *node* yang sedang atau pada lingkungan dengan jumlah *node* 60, 80 dan 100 akan menghasilkan *end-to-end delay* yang lebih bagus daripada di lingkungan dengan jumlah *node* yang jarang maupun yang padat, baik untuk *routing protocol* AODV asli maupun *routing protocol* AODV yang telah dimodifikasi. Ini dikarenakan semakin seimbang perbandingan jumlah *node* dan luas peta uji coba sangat berpengaruh dengan waktu yang dibutuhkan sebuah paket agar dapat mencapai tujuannya. Dengan semakin seimbangnya jumlah *node* dan luas peta uji coba akan memangkas waktu pengiriman dan membuat hasil *end-to-end delay* menjadi lebih baik dari jumlah *node* yang sedikit atau padat, karena dengan jumlah *node* yang sedikit membuat *node* saling menunggu agar dapat terhubung dengan keadaan yang saling menunggu waktu yang diperlukan untuk pengiriman paket pun meningkat sama halnya dengan jumlah *node* yang padat akan menyebabkan *traffic* pengiriman rute semakin padat sehingga akan saling menunggu *node* yang termasuk tujuan paling benar dan menyebabkan waktu pengiriman paket pun semakin lama sehingga membuat hasil *end-to-end delay* dari kedua lingkungan tersebut lebih buruk. Dapat dilihat pula bahwa AODV yang telah dimodifikasi menghasilkan *end-to-end delay* yang lebih bagus daripada AODV asli.

Untuk hasil pengambilan data *routing overhead* pada skenario *node* dengan jumlah *node* 20, 60, 80, 100, 120, 140, 160, 180, dan 200 dapat dilihat pada Gambar 5.5.

Tabel 5.8 Hasil Rata - Rata *Routing Overhead* Skenario *Node*

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**	**Perbedaan (AODV Modifikasi – AODV Asli)**
20	3.70967	3.48489	-0.22478
40	8.21601	7.0873	-1.12871
60	11.26638	10.19693	-1.06945
80	16.0731	15.56577	-0.50733
100	28.28517	20.89193	-7.39324
120	34.99167	30.97221	-4.01946
140	45.68745	34.88512	-10.80233
160	76.9645	56.47921	-20.48529
180	96.03608	67.95121	-28.08487
200	144.26939	105.89836	-38.37103

Dari data pada Tabel 5.8, dibuat grafik yang merepresentasikan hasil perhitungan *routing overhead* yang ditunjukkan pada Gambar 5.6.



Gambar 5.6 Grafik Routing Overhead Skenario *Node*

Berdasarkan grafik pada Gambar 5.6 dapat dilihat bahwa rata-rata *routing overhead* antara *routing protocol* AODV asli dan AODV yang telah dimodifikasi mengalami kenaikan yang signifikan. Pada lingkungan yang jarang dengan jumlah *node* 20, 40, 60 dan 80 menghasilkan perbedaan selisih *routing overhead* masing-masing sebesar 0.22478, 1.12871, 1.06945 dan 0.50733 atau turun masing-masing menjadi sekitar 6.06%, 13.74%, 9.49% dan 3.16% dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam hal *routing overhead* dari AODV asli. Pada lingkungan yang sedang dengan jumlah 100, 120 dan 140 node menghasilkan perbedaan selisih *routing overhead* masing-masing sebesar 7.39324, 4.01946 dan 10.80233 atau turun masing-masing menjadi sekitar 26.14%, 11.49% dan 23.64% dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam hal *routing overhead* dari AODV asli. Pada lingkungan yang padat dengan jumlah 160, 180 dan 200 node menghasilkan perbedaan selisih *routing overhead* masing-masing sebesar 20.4852, 28.08487 dan 38.37103 atau turun masing-masing menjadi sekitar 26.62%, 29.24% dan 26.6% dimana *routing protocol*

AODV yang telah dimodifikasi juga unggul dalam hal *routing overhead* dari AODV asli.

Dapat dilihat rata-rata penurunan yang terjadi untuk *routing overhead* adalah sebesar 17.2%. Jika ketiga lingkungan tersebut dibandingkan dapat dilihat bahwa dengan jumlah *node* yang lebih sedikit atau pada lingkungan dengan *node* yang jarang, menghasilkan *routing overhead* yang lebih bagus atau lebih sedikit daripada di lingkungan dengan jumlah *node* yang sedang maupun yang padat baik untuk *routing protocol* AODV asli maupun *routing protocol* AODV yang telah dimodifikasi. Itu disebabkan dengan jumlah *node* yang sedikit banyak paket yang harus diteruskan baik RREQ, RREP dan RERR akan semakin sedikit dan menyebabkan routing overhead menjadi lebih baik dari pada di lingkungan dengan jumlah *node* yang sedang maupun padat dan karena hal itu jugalah grafik yang dihasilkan akan menaik sejalan dengan jumlah *node* yang diujikan. Dapat dilihat pula bahwa AODV yang telah dimodifikasi menghasilkan *routing overhead* yang lebih sedikit atau dalam hal ini AODV modifikasi lebih unggul di semua lingkungan dengan jumlah selisih *routing overhead* yang cukup signifikan. Hal ini dikarenakan dalam proses pengiriman paket, hanya melewati *reliable node*. Semakin rendah nilai *routing overhead* maka *packet drop* semakin rendah dikarenakan pencarian rute lebih cepat atau optimal dilakukan seiring bertambahnya kepadatan *node*.

5.2.3 Hasil Uji Coba Skenario *Speed*

Pengujian pada skenario *speed* digunakan untuk melihat perbandingan PDR, E2E dan RO antara *routing protocol* AODV asli dan AODV yang telah dimodifikasi dalam pemilihan *node* yang dapat menerima paket *route request*.

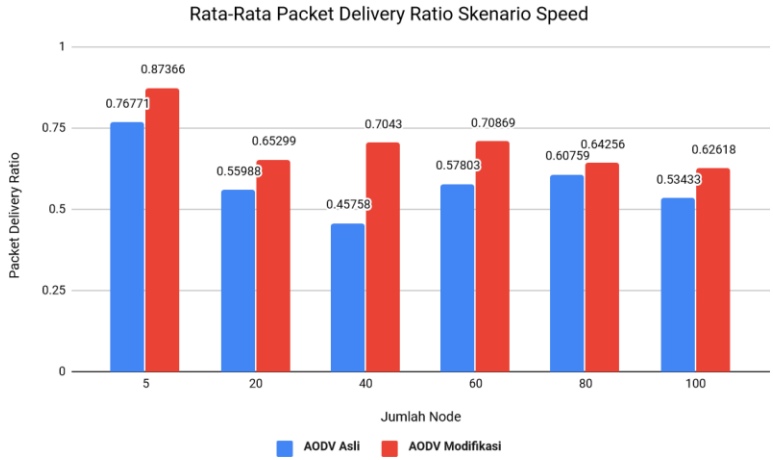
Pengambilan data uji PDR, E2E dan RO pada skenario *real* dilakukan sebanyak 10 kali dengan skenario mobilitas *random* pada skenario *speed* dengan luas area 1000 m x 1000 m dengan *node* sebanyak 100 dan dilakukan pada 5 m/s dan 20 m/s untuk lingkungan kecepatan lambat, 40 m/s dan 60 m/s untuk lingkungan kecepatan

sedang, serta 80 m/s dan 100 m/s untuk lingkungan kecepatan cepat. Untuk hasil analisis skenario *speed* dengan kecepatan 5 m/s, 20 m/s, 60 m/s, 80 m/s dan 100 m/s dapat dilihat pada Tabel 5.6, Tabel 5.7 dan Tabel 5.9.

Tabel 5.9 Hasil Rata - Rata *Packet Delivery Ratio* Skenario *Speed*

Maks *Speed* (m/s)	**AODV Asli**	**AODV Modifikasi**	**Perbedaan (AODV Modifikasi – AODV Asli)**
5	0.76771	0.87366	+ 0.10595
20	0.55988	0.65299	+ 0.09311
40	0.45758	0.7043	+ 0.24672
60	0.57803	0.70869	+ 0.13066
80	0.60759	0.64256	+ 0.03497
100	0.53433	0.62618	+ 0.09185

Dari data pada Tabel 5.9, dibuat grafik yang merepresentasikan hasil perhitungan PDR yang ditunjukkan pada Gambar 5.7.



Gambar 5.7 Grafik *Packet Delivery Ratio* Skenario *Speed*

Berdasarkan grafik pada Gambar 5.7 dapat dilihat bahwa *routing protocol* AODV yang telah dimodifikasi dan juga *routing protocol* AODV asli mengalami kenaikan yang signifikan pada *packet delivery ratio*. Pada lingkungan dengan kecepatan maksimum acak 5 m/s dan 20 m/s menghasilkan perbedaan selisih PDR masing-masing sebesar 0.10595 dan 0.09311 atau naik masing-masing menjadi sekitar 13.8% dan 16.63% dimana PDR AODV yang telah dimodifikasi unggul dalam hal PDR dari AODV asli. Pada lingkungan dengan kecepatan maksimum acak 40 m/s dan 60 m/s menghasilkan perbedaan selisih PDR masing-masing sebesar 0.24672 dan 0.13066 atau naik masing-masing menjadi sekitar 53.92% dan 22.6% dimana PDR AODV yang telah dimodifikasi unggul dalam hal PDR dari AODV asli. Pada lingkungan dengan kecepatan maksimum acak 80 m/s dan 100 m/s menghasilkan perbedaan selisih PDR masing-masing sebesar 0.03497 dan 0.09185 atau naik masing-masing menjadi sekitar 5.76% dan 17.19% dimana PDR AODV yang telah dimodifikasi juga unggul dalam hal PDR dari AODV asli.

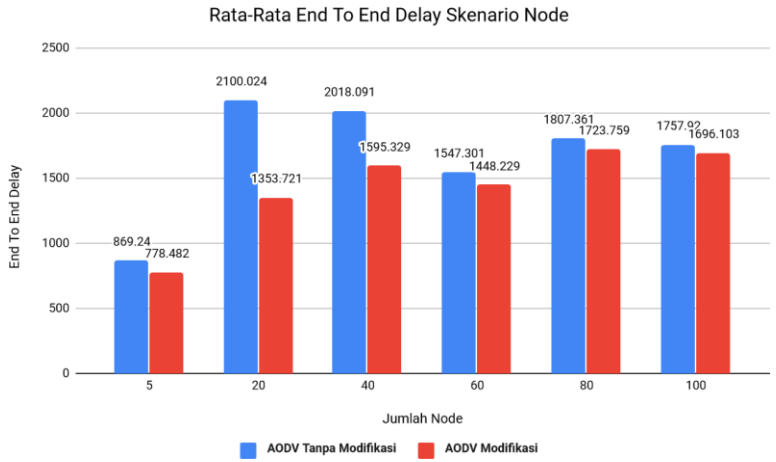
Dapat dilihat rata-rata kenaikan yang terjadi untuk PDR adalah 21.65%. Dapat dilihat juga pada AODV modifikasi hasil PDR tergolong lebih stabil, meski dapat kita menarik kesimpulan dengan kecepatan maksimum acak yang stabil pada titik 5 m/s PDR yang dihasilkan akan sangat bagus, baik di AODV modifikasi ataupun AODV tanpa modifikasi. Itu disebabkan tidak banyaknya pilihan kecepatan yang dipunyai oleh setiap *node* dan lingkungan uji coba hanya akan membuat semua *node* menghasilkan sudut θ yang sama dan digolongkan menjadi *reliable node*. Disini kita juga dapat menyimpulkan bahwa AODV yang telah dimodifikasi menghasilkan PDR yang lebih bagus daripada AODV asli dengan jumlah selisih PDR yang cukup signifikan.

Hasil pengambilan data rata-rata untuk *end-to-end delay* (E2E) pada skenario *speed* dengan kecepatan 5 m/s, 20 m/s, 60 m/s, 80 m/s dan 100 m/s dapat dilihat pada Tabel 5.10.

Tabel 5.10 Hasil Rata -Rata End-To-End Delay Skenario *Speed*

Maks *Speed* (m/s)	**AODV Asli**	**AODV Modifikasi**	**Perbedaan (AODV Modifikasi – AODV Asli)**
5	869.24	778.482	-90.758
20	2100.024	1353.721	-746.303
40	2018.091	1595.329	-422.762
60	1547.301	1448.229	-99.072
80	1807.361	1723.759	-83.602
100	1757.92	1696.103	-61.817

Dari data pada Tabel 5.10, dibuat grafik yang merepresentasikan hasil perhitungan *end-to-end delay* yang ditunjukkan pada Gambar 5.8.



Gambar 5.8 Grafik *End-to-end Delay* pada Skenario *Speed*

Berdasarkan grafik pada Gambar 5.8 dapat dilihat bahwa rata-rata *end-to-end delay* antara *routing protocol* AODV asli dan

AODV yang telah dimodifikasi mengalami penurunan yang cukup signifikan, tetapi pada kecepatan maksimum 60 m/s dan 100 m/s *end-to-end delay* mengalami kenaikan Pada lingkungan dengan kecepatan maksimum acak 5 m/s dan 20 m/s menghasilkan perbedaan selisih *end-to-end delay* masing-masing sebesar 90.758 dan 746.30 atau turun masing-masing menjadi sekitar 10.44% dan 35.54% dimana *end-to-end delay* AODV yang telah dimodifikasi unggul dalam hal *end-to-end delay* dari AODV asli. Pada lingkungan dengan kecepatan maksimum acak 40 m/s dan 60 m/s menghasilkan perbedaan selisih *end-to-end delay* masing-masing sebesar 422.76 dan 99.072 atau turun masing-masing menjadi sekitar 20,94% dan 6.40% dimana *end-to-end delay* AODV yang telah dimodifikasi unggul dalam hal *end-to-end delay* dari AODV asli. Pada lingkungan dengan kecepatan maksimum acak 80 m/s dan 100 m/s menghasilkan perbedaan selisih *end-to-end delay* masing-masing sebesar 83.602 dan 61.817 atau turun masing-masing menjadi sekitar 4,63% dan 3.52% dimana *end-to-end delay* AODV yang telah dimodifikasi juga unggul dalam hal *end-to-end delay* dari AODV asli.

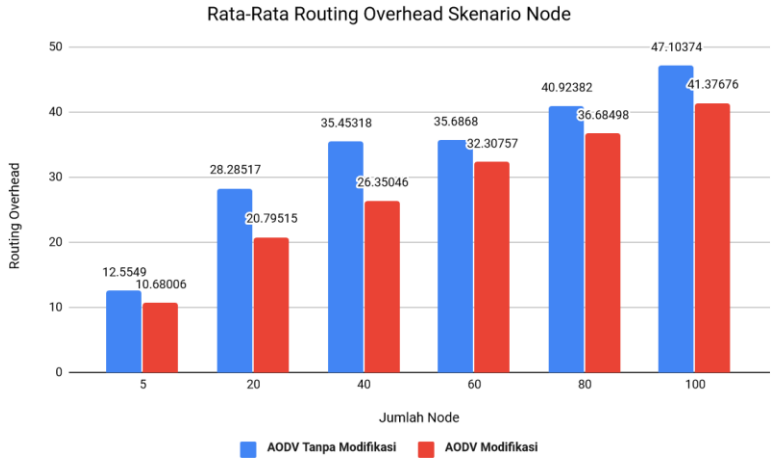
Dapat dilihat rata-rata penurunan yang terjadi untuk *end-to-end delay* adalah 13,58%. Dapat dilihat juga pada AODV modifikasi hasil *end-to-end delay* tergolong lebih stabil, meski dapat kita menarik kesimpulan dengan kecepatan maksimum acak yang stabil pada titik 5 m/s *end-to-end delay* yang dihasilkan akan sangat bagus, baik di AODV modifikasi ataupun AODV tanpa modifikasi. Itu disebabkan tidak banyaknya pilihan kecepatan yang dipunyai oleh setiap *node* dan lingkungan uji coba hanya akan membuat semua *node* menghasilkan sudut θ yang sama dan digolongkan menjadi *reliable node*. Disini kita juga dapat menyimpulkan bahwa AODV yang telah dimodifikasi menghasilkan *end-to-end delay* yang lebih bagus daripada AODV asli dengan jumlah selisih *end-to-end delay* yang cukup signifikan.

Untuk hasil pengambilan data *routing overhead* pada skenario *speed* dengan kecepatan 5 m/s, 20 m/s, 60 m/s, 80 m/s dan 100 m/s dapat dilihat pada Tabel 5.11.

Tabel 5.11 Hasil Rata - Rata *Routing Overhead* Skenario *Speed*

Maks *Speed* (m/s)	AODV Asli	AODV Modifikasi	Perbedaan (AODV Modifikasi – AODV Asli)
5	12.5549	10.68006	-1.87484
20	28.28517	20.79515	-7.49002
40	35.45318	26.35046	-9.10272
60	35.6868	32.30757	-3.37923
80	40.92382	36.68498	-4.23884
100	47.10374	41.37676	-5.72698

Dari data pada Tabel 5.11, dibuat grafik yang merepresentasikan hasil perhitungan *routing overhead* yang ditunjukkan pada Gambar 5.9.



Gambar 5.9 Grafik *Routing Overhead* Skenario *Speed*

Berdasarkan grafik pada Gambar 5.9 dapat dilihat bahwa rata-rata *routing overhead* antara *routing protocol* AODV asli dan AODV yang telah dimodifikasi mengalami penurunan yang signifikan. Pada lingkungan dengan kecepatan maksimum acak 5 m/s

dan 20 m/s menghasilkan perbedaan selisih *routing overhead* masing-masing sebesar 1.87484 dan 7.49002 atau turun masing-masing menjadi sekitar 14.93% dan 26.48% dimana *routing overhead* AODV yang telah dimodifikasi unggul dalam hal *routing overhead* dari AODV asli. Pada lingkungan dengan kecepatan maksimum acak 40 m/s dan 60 m/s menghasilkan perbedaan selisih *routing overhead* masing-masing sebesar 9.10272 dan 3.37923 atau naik masing-masing menjadi sekitar 25.68% dan 9.47% dimana *routing overhead* AODV yang telah dimodifikasi unggul dalam hal *routing overhead* dari AODV asli. Pada lingkungan dengan kecepatan maksimum acak 80 m/s dan 100 m/s menghasilkan perbedaan selisih *routing overhead* masing-masing sebesar 4.23884 dan 5.72698 atau naik masing-masing menjadi sekitar 10.36% dan 12.16% dimana *routing overhead* AODV yang telah dimodifikasi juga unggul dalam hal *routing overhead* dari AODV asli.

Dapat dilihat rata-rata penurunan yang terjadi untuk *routing overhead* adalah sebesar 16.51%. Dapat dilihat juga pada AODV modifikasi hasil *routing overhead* tergolong lebih stabil, meski dapat kita menarik kesimpulan dengan kecepatan maksimum acak yang stabil pada titik 5 m/s *routing overhead* yang dihasilkan akan sangat bagus, baik di AODV modifikasi ataupun AODV tanpa modifikasi. Itu disebabkan tidak banyaknya pilihan kecepatan yang dipunyai oleh setiap *node* dan lingkungan uji coba hanya akan membuat semua *node* menghasilkan sudut θ yang sama dan digolongkan menjadi *reliable node*. Disini kita juga dapat menyimpulkan bahwa AODV yang telah dimodifikasi menghasilkan *routing overhead* yang lebih sedikit atau dalam hal ini AODV modifkikasi lebih unggul di semua lingkungan dengan jumlah selisih *routing overhead* yang cukup signifikan.

(Halaman ini sengaja dikosongkan)

BAB VI
KESIMPULAN DAN SARAN

Pada Bab ini akan diberikan kesimpulan yang diperoleh dari Tugas Akhir yang telah dikerjakan dan saran tentang pengembangan dari Tugas Akhir ini yang dapat dilakukan di masa yang akan datang.

6.1 Kesimpulan

Kesimpulan yang diperoleh pada uji coba dan evaluasi Tugas Akhir ini adalah sebagai berikut:

1. Dengan menggunakan metode *velocity-aware*. *Node* dapat dibagi menjadi dua jenis. Pembagian ini dilakukan dengan cara menghitung sudut θ yang dihasilkan oleh kecepatan *node* pengirim dan *node* penerima apakah kurang dari *threshold* yang sudah diberikan atau tidak. Apabila modifikasi sudah dilakukan maka hasil *routing overhead* akan berkurang dikarenakan *reliable node* terpilih sebagai pengantar paket yang membuat kemungkinan RERR terjadi pada AODV modifikasi semakin kecil, juga membuat jumlah RREP serta RREQ berkurang dikarenakan rute yang semakin pendek. Hal ini dapat dibuktikan dengan hasil *routing overhead* yang berkurang pada dua skenario uji coba yang telah dilakukan. Uji coba di skenario *node* rata-rata menurunkan *routing overhead* sebesar 17,62%, sedangkan pada skenario *speed* rata-rata menurunkan *routing overhead* sebesar 16,51%.
2. Dampak metode *velocity-aware* terhadap performa protokol AODV pada skenario *node* atau dengan jumlah *node* yang berbeda-beda memiliki rata – rata kenaikan *Packet Delivery Ratio* sebesar 21,34%, rata – rata penurunan *End-To-End Delay* sebesar 10,45% dan rata – rata penurunan *Routing Overhead* sebesar 17,62%. Sedangkan dampak metode *velocity-aware* terhadap performa protokol AODV pada skenario *speed* adalah rata – rata kenaikan *Packet Delivery Ratio* sebesar 21,65%, rata

– rata penurunan *End-To-End Delay* sebesar 13,8% dan rata – rata penurunan *Routing Overhead* sebesar 16,51%.

6.2 Saran

Saran yang dapat diberikan dari hasil uji coba dan evaluasi adalah sebagai berikut:

1. Membuat *threshold* menjadi adaptif, sehingga apabila rute tidak ditemukan dapat menaikkan *threshold* yang sudah diberikan, serta apabila rute banyak yang berhasil ditemukan atau terjadi *broken link* maka dapat menurunkan *threshold* yang sudah ada.
2. *Threshold* adaptif yang dimaksudkan adalah *threshold* yang bisa menaikkan dan menurunkan ambang batas sudut secara otomatis tergantung akan keadaan lingkungan pengujian, contoh menaikkan *threshold* dari 165 menjadi 180 bila tidak ditemukan rute, serta menurunkan *threshold* dari 165 menjadi 150 bila terlalu banyak menemukan rute.

DAFTAR PUSTAKA

[1] Y. Feng , B. Zhang, S. Chai, L. Cui dan Q. Li, "An Optimized AODV Protocol based on Clustering for WSNs," *6th International Conference on Computer Science and Network Technology (ICCSNT),* 2017.

[2] R. Brendha dan V. S. J. Prakash, "A Survey on Routing Protocols for Vehicular Ad hoc Networks," IEEE, Coimbatore, 2017.

[3] R. F. Sari dan A. Syarif, "Analisis Kinerja Protokol Routing Ad Hoc On-Demand Distance Vector (AODV) pada Jaringan Ad Hoc," p. 22, October 2010.

[4] P. Meeneghan dan D. Delaney, "An Introduction to NS Nam and OTcl scripting," April 2004.

[5] "OpenStreetMap," [Online]. Available: https://www.openstreetmap.org/. [Diakses 18 Desember 2018].

[6] "AWK," [Online]. Available: http://tldp.org/LDP/abs/html/awk.html. [Diakses 20 Desember 2018].

[7] M. Iqbal, M. Shafiq, H. Attaullah, J.-G. Choi, K. Akram dan X. Wang, "Design and Analysis of a Novel Hybrid Wireless Mesh Network Routing Protocol," p. 22, January 2014.

[8] Gupta, S. K., & Saket, R. K. (2011). PERFORMANCE METRIC COMPARISON OF AODV AND DSDV ROUTING PROTOCOLS IN MANETs USING NS-2 ., (pp. 344-345). Varanasi.

[9] Farooq, M. U., & Tapus, N. (2014). Multiclass On-demand Routing in Heterogeneous Ad hoc Networks.

[10] Moore, D. et. al. (2001, June 12). *Bluetooth Network Encapsulation Protocol (BNEP) Specification.* Retrieved from http://grouper.ieee.org/groups/802/15/Bluetooth/BNEP.pdf

[11] Mustafa Bani Khalaf, Ahmed Y. Al-Dubai and Mourad Abed. (2012, Sept 11).New Velocity Aware Probabilistic Route Discovery Schemes for Mobile Ad hoc Networks from https://ieeexplore.ieee.org/document/6347633.

[12] Simamora, S.N.M.P., 2014. Model Pembelajaran Teknologi Informasi Dengan Teknik MANET Pada Kawasan Tertinggal. SENANTI (Prosiding Seminar Nasional Indonesia Timur), 14 Juni 2014. Yogyakarta: Universitas Atma Jaya Yogyakarta.

[13] Elboukhori, M., Azizi, M., & Azizi, A., 2015. Performance Comparison of Protokol routings in Mobile Ad Hoc Networks. International Journal of UbiComp (IJU), 6(2), p.3.

[14] Septian, S., 2015. Pengujian Penerapan Teknologi Jaringan Mobile Ad-hoc Network (MANET) untuk File Transfer. [Online] Tersedia di: < http://sigitseptian.student.telkomuniversity.ac.id/contoh-implementasi-vanet-danmanet/> [Diakses 15 Februari 2018]

[15] Alfinanto, A., 2016. Analisis Perbandingan Unjuk Kerja Protokol Routing Reactive (DYMO) Terhadap Protokol Routing Reactive (AODV) Pada Jaringan MANET. S1. Universitas Sanata Dharma. Tersedia di < https://repository.usd.ac.id/9009/2/125314077_full.pdf> [Diakses 15 Februari 2018]

LAMPIRAN

A.1 Kode Aodv Modifikasi

```
//Mendapatkan kecepatan pengirim RREQ
double sendervx, sendervy, sendervz;
Node* sender = Node::get_node_by_address(ih-
>saddr());
((MobileNode *)sender)->getVelo(&sendervx,
&sendervy, &sendervz);

//Mendapatkan kecepatan penerima RREQ
double receivervx, receivervy, receivervz;
Node* receiver =
Node::get_node_by_address(index);
((MobileNode *)receiver)->getVelo(&receivervx,
&receivervy, &receivervz);

//Agar tidak error ketika dibagi 0
if(sendervx==0) sendervx=0.0001;
if(sendervy==0) sendervy=0.0001;
if(receivervx==0) receivervx=0.0001;
if(receivervy==0) receivervy=0.0001;

//rumus sudut θ
double PI= 3.14159265;
double val = 180.0 / PI;
double pembilang = sendervx * receivervx +
sendervy * receivervy;
double penyebut =
sqrt(pow(sendervx,2)+pow(sendervy,2))*sqrt(pow
(receivervx,2)+pow(receivervy,2));
double teta=acos(pembilang/penyebut)*val;
```

```
//mengecek threshold
double tetathreshold=165;
if(teta > tetathreshold){
	Packet::free(p); //drop packet
	return;
}
```

A.2 Kode Skenario NS-2

```
set val(chan)   Channel/WirelessChannel;
set val(prop)   Propagation/TwoRayGround;
set val(netif)  Phy/WirelessPhy;
set val(mac)    Mac/802_11;
set val(ifq)    Queue/DropTail/PriQueue;
set val(ll)     LL;
set val(ant)    Antenna/OmniAntenna;
set opt(x)      1000;
set opt(y)      1000;
set val(ifqlen) 50;
set val(nn)     20;
set val(seed)   1.0;
set val(adhocRouting)   AODV;
set val(stop)   200;
set val(cp)     "cbr.tcl";
set val(sc)     "setdest20.tcl";

set ns_         [new Simulator]

# setup topography object
set topo   [new Topography]

# create trace object for ns and nam
set tracefd     [open result/result.tr w]
set namtrace    [open result/result.nam
w]

$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace
$opt(x) $opt(y)

# set up topology object
set topo        [new Topography]
$topo load_flatgrid $opt(x) $opt(y)
```

```
# Create God
set god_ [create-god $val(nn)]

#global node setting

#no energy
 $ns_ node-config    -adhocRouting
$val(adhocRouting) \
                     -llType $val(ll) \
                     -macType $val(mac) \
                     -ifqType $val(ifq) \
                     -ifqLen $val(ifqlen)
\
                     -antType $val(ant) \
                     -propType $val(prop)
\
                     -phyType $val(netif)
\
                     -channelType
$val(chan) \
                     -topoInstance $topo
\
                     -agentTrace ON \
                     -routerTrace ON \
                     -macTrace ON \
                     -movementTrace ON

Phy/WirelessPhy set   RXThresh_ 5.57189e-
11 ; #400m receiver sensitivity range
Phy/WirelessPhy set   CSThresh_ 5.57189e-
11 ; #400m capture threshold  range

for {set i 0} {$i < $val(nn)} {incr i} {
    set node_($i) [$ns_ node]
    $node_($i) random-motion 0 ;# disable
random motion
}
```

```
# Define node movement model
puts "Loading connection pattern..."
source $val(cp)

# Define traffic model
puts "Loading scenario file..."
source $val(sc)

# Define node initial position in nam

for {set i 0} {$i < $val(nn)} {incr i} {

    # 20 defines the node size in nam,
must adjust it according to your scenario
    # The function must be called after
mobility model is defined

    $ns_ initial_node_pos $node_($i) 20
}

# Tell nodes when the simulation ends
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $val(stop).0 "$node_($i)
reset";
}

#$ns_ at  $val(stop)  "stop"
$ns_ at  $val(stop).0002 "puts \"NS
EXITING...\" ; $ns_ halt"

puts $tracefd "M 0.0 nn $val(nn) x
$opt(x) y $opt(y) rp $val(adhocRouting)"
puts $tracefd "M 0.0 sc $val(sc) cp
$val(cp) seed $val(seed)"
puts $tracefd "M 0.0 prop $val(prop) ant
$val(ant)"

puts "Starting Simulation..."
$ns_ run
```

A.3 Kode Konfigurasi *Traffic*

```
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(18) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(19) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_
0.20000000000000001
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 2.5568388786897245 "$cbr_(0)
start"
```

A.4 Kode Skrip AWK *Packet Delivery Ratio*

```
BEGIN {
        sendLine = 0;
        recvLine = 0;
        fowardLine = 0;
}

$0 ~/^s.* AGT/ {
        sendLine ++ ;
}

$0 ~/^r.* AGT/ {
        recvLine ++ ;
}

$0 ~/^f.* RTR/ {
        fowardLine ++ ;
}

END {
        printf "cbr s:%d r:%d, r/s
Ratio:%.4f, f:%d \n", sendLine, recvLine,
(recvLine/sendLine),fowardLine;
}
```

A.5 Kode Skrip AWK Rata-Rata *End-to-End Delay*

```
BEGIN{
      sum_delay = 0;
      count = 0;
}
{
      if ($2 >= 101) {

            if($4 == "AGT" && $1 == "s" &&
seqno < $6) {
              seqno = $6;
            }

            if($4 == "AGT" && $1 == "s") {
                  start_time[$6] = $2;
            }

            else if(($7 == "cbr") && ($1
== "r")) {
                  end_time[$6] = $2;
            }

            else if($1 == "D" && $7 ==
"cbr") {
                  end_time[$6] = -1;
            }
      }
}
END {
    for(i=0; i<=seqno; i++) {
            if(end_time[i] > 0) {
                  delay[i] = end_time[i] -
```

```
start_time[i];
                count++;
        }
        else {
           delay[i] = -1;
        }
    }
    for(i=0; i<=seqno; i++) {
        if(delay[i] > 0) {
           n_to_n_delay = n_to_n_delay +
delay[i];
        }
    }
     n_to_n_delay = n_to_n_delay/count;
     printf "End-to-End Delay \t= "
n_to_n_delay * 1000 " ms \n";
}
```

A.6 Kode Skrip AWK *Routing Overhead*

```
BEGIN {
     rt_pkts = 0;
}
{
           if (($1 == "s" || $1 == "f")
&& ($4 == "RTR") && ($7 == "AODV")) {

                  rt_pkts++;
           }
     }
}
END {
     printf "Routing Packets \t= %d \n",
rt_pkts;
}
```

B.1 Tabel Hasil Skenario *Threshold* Terbaik 0º

MODIFIKASI							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	995	0	995	0	0	0	inf
2	996	0	996	0	0	0	inf
3	991	0	991	0	0	0	inf
4	995	0	995	0	0	0	inf
5	992	0	992	0	0	0	inf
6	994	0	994	0	0	0	inf
7	992	0	992	0	0	0	inf
8	993	0	993	0	0	0	inf
9	990	0	990	0	0	0	inf
10	985	0	985	0	0	0	inf

Jenis	PDR	E2E	RO
Modifikasi	0	0	0

B.2 Tabel Hasil Skenario *Threshold* Terbaik 33º

MODIFIKASI							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	982	549	433	0.5591	2026.71	3311	26.776
2	998	302	696	0.3026	2315.54	1608	32.1126
3	987	475	512	0.4813	2546.4	2228	21.8821
4	977	171	806	0.175	3307.86	779	40.924
5	976	349	627	0.3576	1806.56	1796	31.2894
6	1012	347	665	0.3429	2141.69	1907	30.7839
7	985	413	572	0.4193	2223.54	1828	18.1598
8	979	435	544	0.4443	884.22	2296	24.4414
9	973	483	490	0.4964	1111.7	2324	19.0559
10	989	541	448	0.547	1285.72	2692	15.3937

Jenis	PDR	E2E	RO
Modifikasi	0.41255	1964.994	26.08188

B.3 Tabel Hasil Skenario *Threshold* Terbaik 66º

MODIFIKASI							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	978	381	597	0.3896	3354.71	1977	33.0604
2	983	384	599	0.3906	1603.88	1830	31.9115
3	979	490	489	0.5005	1386.42	2175	25.1878
4	989	471	518	0.4762	1676.3	2409	25.0488
5	974	577	397	0.5924	1797.26	2765	24.3917
6	989	556	433	0.5622	1772.03	2645	22.964
7	978	600	378	0.6135	2058.74	2956	18.5233
8	978	508	470	0.5194	1141.32	2470	23.9134
9	979	616	363	0.6292	2270.31	3021	17.4935
10	992	499	493	0.503	2209.82	2173	19.5631

Jenis	PDR	E2E	RO
Modifikasi	0.51766	1927.079	24.20575

B.4 Tabel Hasil Skenario *Threshold* Terbaik 99º

MODIFIKASI							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	991	517	474	0.5217	2435.78	2516	22.8975
2	978	777	201	0.7945	560.81	3699	17.2999
3	1001	632	369	0.6314	1968.01	3080	23.8386
4	984	510	474	0.5183	2027.69	2050	18.3098
5	987	495	492	0.5015	1535.07	2297	22.5576
6	982	442	540	0.4501	2894.96	2168	30.371
7	985	666	319	0.6761	1276.98	3343	17.9459
8	964	568	396	0.5892	1888.9	2585	25.9331
9	991	562	429	0.5671	2318.17	2510	24.5765
10	985	738	247	0.7492	1418.05	3459	18.8455

Jenis	PDR	E2E	RO
Modifikasi	0.59991	1832.442	22.25754

B.5 Tabel Hasil Skenario *Threshold* Terbaik 132º

MODIFIKASI							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	991	517	474	0.5217	2435.78	2516	22.8975
2	978	777	201	0.7945	560.81	3699	17.2999
3	1001	632	369	0.6314	1968.01	3080	23.8386
4	984	510	474	0.5183	2027.69	2050	18.3098
5	987	495	492	0.5015	1535.07	2297	22.5576
6	982	442	540	0.4501	2894.96	2168	30.371
7	985	666	319	0.6761	1276.98	3343	17.9459
8	964	568	396	0.5892	1888.9	2585	25.9331
9	991	562	429	0.5671	2318.17	2510	24.5765
10	985	738	247	0.7492	1418.05	3459	18.8455

Jenis	PDR	E2E	RO
Modifikasi	0.67495	1473.696	21.00204

B.6 Tabel Hasil Skenario *Threshold* Terbaik 165º

MODIFIKASI							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	991	517	474	0.5217	2435.78	2516	22.8975
2	978	777	201	0.7945	560.81	3699	17.2999
3	1001	632	369	0.6314	1968.01	3080	23.8386
4	984	510	474	0.5183	2027.69	2050	18.3098
5	987	495	492	0.5015	1535.07	2297	22.5576
6	982	442	540	0.4501	2894.96	2168	30.371
7	985	666	319	0.6761	1276.98	3343	17.9459
8	964	568	396	0.5892	1888.9	2585	25.9331
9	991	562	429	0.5671	2318.17	2510	24.5765
10	985	738	247	0.7492	1418.05	3459	18.8455

Jenis	PDR	E2E	RO
Modifikasi	0.76832	1338.234	20.89193

B.7 Tabel Hasil Skenario *Node* 20 *node*

MODIFIKASI							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	987	583	404	0.5907	1620.61	3090	4.4425
2	981	434	547	0.4424	1001.38	2317	3.8249
3	983	589	394	0.5992	1508.73	3244	3.871
4	975	542	433	0.5559	1559.45	2893	2.428
5	1002	623	379	0.6218	672.3	2995	2.2183
6	991	676	315	0.6821	1096.79	3012	3.3876
7	979	493	486	0.5036	1113.48	2562	3.0507
8	987	598	389	0.6059	1291.07	3097	2.806
9	994	710	284	0.7143	865.41	3829	4.0141
10	984	551	433	0.56	1305.92	3017	4.8058

NORMAL							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	970	735	235	0.7577	1252.51	3502	3.1646
2	972	688	284	0.7078	840.16	3581	2.936
3	988	492	496	0.498	1820.32	2779	5.0732
4	983	453	530	0.4608	1529.76	2600	4.2649
5	984	680	304	0.6911	885.72	3122	2.5647

6	979	455	524	0.4648	1208.77	1993	3.7319
7	993	314	679	0.3162	1391.72	1600	4.2866
8	995	385	610	0.3869	958.45	2139	3.761
9	977	577	400	0.5906	809.97	3490	4.0069
10	983	580	403	0.59	1583.18	3040	3.3069

Jenis	PDR	E2E	RO
Modifikasi	0.58759	1203.514	3.48489
Original	0.54639	1228.056	3.70967
Mod - Ori	0.0412	-24.542	-0.22478

B.8 Tabel Hasil Skenario *Node* 40 *node*

MODIFIKASI							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	969	685	284	0.7069	1215.87	3376	8.3679
2	995	637	358	0.6402	1304.99	3586	7.529
3	989	662	327	0.6694	1466.02	3456	8.9789
4	973	908	65	0.9332	454.67	4670	4.2423
5	982	670	312	0.6823	1492.43	3093	7.7672
6	976	701	275	0.7182	1017.69	3467	7.9914
7	990	775	215	0.7828	1576.66	4173	6.8748
8	977	698	279	0.7144	1242.28	3859	7.596

9	990	755	235	0.7626	667.95	3640	6.2596
10	978	801	177	0.819	569.73	3882	5.2659

NORMAL							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	989	723	266	0.731	799.85	3322	5.2531
2	987	788	199	0.7984	689.71	3915	7.802
3	992	297	695	0.2994	1982.89	1778	17.5219
4	973	808	165	0.8304	1764.06	4379	7.3243
5	990	673	317	0.6798	2017.51	2999	7.6256
6	988	647	341	0.6549	1101.98	3537	8.4297
7	988	780	208	0.7895	732.82	5144	6.1462
8	992	703	289	0.7087	477.31	3647	8.697
9	989	791	198	0.7998	669.2	3644	5.0746
10	972	777	195	0.7994	1095.36	3923	8.2857

Jenis	PDR	E2E	RO
Modifikasi	0.7429	1100.829	7.0873
Original	0.70913	1133.069	8.21601
Mod - Ori	0.03377	-32.24	-1.12871

B.9 Tabel Hasil Skenario *Node* 60 *node*

MODIFIKASI							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	986	759	227	0.7698	1021.85	3512	10.6324
2	973	749	224	0.7698	1160.37	4341	9.3324
3	978	733	245	0.7495	1023.97	3512	7.1596
4	990	851	139	0.8596	939.8	4168	9.4383
5	983	960	23	0.9766	877.25	4542	9.5396
6	993	765	228	0.7704	1080.11	3557	10.7425
7	979	739	240	0.7549	1195.51	3449	15.8187
8	990	866	124	0.8747	1151.22	4158	8.8915
9	986	901	85	0.9138	537.48	4192	7.7558
10	990	814	176	0.8222	799.97	4341	12.6585

NORMAL							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	974	948	26	0.9733	591.66	4158	7.7321
2	982	590	392	0.6008	1780.34	2567	11.7254
3	988	887	101	0.8978	947.29	4280	9.5603
4	967	838	129	0.8666	971.68	4483	12.043
5	987	883	104	0.8946	861.55	4416	10.6954

6	971	892	79	0.9186	676.17	3767	10.213
7	985	854	131	0.867	776.88	4291	10.0422
8	982	803	179	0.8177	833.91	3722	12.4134
9	1000	464	536	0.464	1875.82	2239	14.431
10	967	927	40	0.8586	919.92	4483	13.808

Jenis	PDR	E2E	RO
Modifikasi	0.82613	978.753	10.19693
Original	0.8159	1023.522	11.26638
Mod - Ori	0.01023	-44.769	-1.06945

B.10 Tabel Hasil Skenario *Node* 80 *node*

MODIFIKASI							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	989	794	195	0.8028	289.54	3780	17.204
2	983	746	237	0.7589	1369.71	3242	15.4263
3	1003	912	91	0.9093	900.52	4360	17.3048
4	988	727	261	0.7358	1304.98	3139	15.3673
5	978	929	49	0.9499	696.11	4433	12.3251
6	972	803	169	0.8261	1093.91	3992	15.4869
7	989	769	220	0.7776	1199.76	3859	21.1521
8	980	827	153	0.8439	792.65	3761	13.4172

9	979	943	36	0.9632	432.61	4362	15.0053
10	999	799	200	0.7998	864.85	3644	12.9687

NORMAL							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	975	800	175	0.8205	383.03	3883	12.675
2	976	824	152	0.8443	681.87	3760	17.0121
3	978	713	265	0.729	1550.9	3574	17.5568
4	988	900	88	0.9109	404.43	4195	11.5622
5	1002	345	657	0.3443	894.85	1857	23.7217
6	997	701	296	0.7031	1315.03	3479	15.8402
7	977	813	164	0.8321	743.79	4154	13.6052
8	972	742	230	0.7634	496.34	3638	15.4582
9	971	907	64	0.9341	468.52	3965	10.0088
10	975	423	552	0.4338	3701.99	2035	23.2908

Jenis	PDR	E2E	RO
Modifikasi	0.83673	894.464	15.56577
Original	0.73155	1064.075	16.0731
Mod - Ori	0.10518	-169.611	-0.50733

B.11 Tabel Hasil Skenario *Node* 100 *node*

MODIFIKASI							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	1001	697	304	0.6963	2381.17	3270	25.5983
2	978	569	409	0.5818	1352.27	2460	18.2425
3	988	701	287	0.7095	2453.66	3674	30.4765
4	993	852	141	0.858	946.93	4019	20.9319
5	1000	791	209	0.791	1670.63	3623	17.7446
6	986	612	374	0.6207	1577.92	2817	17.915
7	980	880	100	0.898	845.87	4195	12.8523
8	986	960	26	0.9736	238.21	4599	14.5604
9	986	899	87	0.9118	928.41	4195	27.7953
10	993	638	355	0.6425	987.27	3482	22.8025

NORMAL							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	982	388	594	0.3951	2016.09	2044	30.0825
2	983	531	452	0.5402	1318.87	2591	17.484
3	992	667	325	0.6724	1401.13	2992	25.7151
4	969	840	129	0.8669	726.8	4102	23.5667
5	989	548	441	0.5541	1813.83	2571	28.1898

6	994	581	413	0.5845	1701.93	2715	33.043
7	980	596	384	0.6082	1725.98	2662	24.1913
8	1000	725	275	0.725	1249.66	3333	26.8083
9	983	385	598	0.3917	2154.85	1777	35.8494
10	978	255	723	0.2607	2591.1	1265	37.9216

Jenis	PDR	E2E	RO
Modifikasi	0.76832	1338.234	20.89193
Original	0.55988	1670.024	28.28517
Mod - Ori	0.20844	-331.79	-7.39324

B.12 Tabel Hasil Skenario *Node* 120 *node*

MODIFIKASI							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	984	919	65	0.9339	751.99	4470	25.2427
2	993	577	416	0.5811	1449.56	2888	31.0919
3	994	667	327	0.671	825.25	3002	26.2429
4	994	495	499	0.498	3106.66	2233	41.0949
5	1000	765	235	0.765	1768.59	3367	27.4484
6	995	557	438	0.5598	2373.68	2736	39.6948
7	978	453	525	0.4632	2754.28	2290	46.7329
8	994	481	513	0.4839	2078.31	2223	37.1767

9	990	721	269	0.7283	1662.47	3636	34.982
10	988	507	481	0.5132	2635.57	2470	39.144

NORMAL							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	976	553	423	0.5666	1396.67	3075	27.1971
2	990	599	391	0.6051	1388.69	2960	27.025
3	978	771	207	0.7883	1472.32	3673	30.7808
4	1000	155	845	0.155	5174.48	855	91.1355
5	993	314	679	0.3162	1713.06	1594	47.707
6	989	263	726	0.2659	3298.15	1179	40
7	992	285	707	0.2873	3447.74	1229	65.5579
8	995	587	408	0.5899	2065.23	2608	41.5639
9	974	510	464	0.5236	1820.28	2494	39.2667
10	980	537	443	0.548	2986.3	2964	46.6406

Jenis	PDR	E2E	RO
Modifikasi	0.61974	1940.636	34.88512
Original	0.46459	2476.292	45.68745
Mod - Ori	0.15515	-535.656	-10.80233

B.13 Tabel Hasil Skenario *Node* 140 *node*

MODIFIKASI							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	978	429	549	0.4387	787.84	1888	41.1189
2	980	309	671	0.3153	2155.48	1408	84.6731
3	990	370	620	0.3737	2262.74	1615	53.2432
4	979	253	726	0.2584	3359.86	1185	60.0079
5	982	360	622	0.3666	3439.01	1688	59.0444
6	989	389	600	0.3933	3061.44	1646	46.6787
7	981	335	646	0.3415	2657.64	1779	51.9403
8	997	430	567	0.4313	1991.56	2227	51.8419
9	1007	283	724	0.281	2893.42	1334	60.2261
10	974	341	633	0.3501	2500.54	1514	56.0176

NORMAL							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	975	197	778	0.2021	1815.68	948	115.0558
2	989	399	590	0.4034	1709.8	1738	55.2281
3	979	131	848	0.1338	4721.26	640	112.5802
4	990	265	725	0.2677	1976.99	1332	67.0943
5	973	39	934	0.0401	3069.91	209	121.5385

6	986	192	794	0.1947	4432.21	997	103.2292
7	989	577	412	0.5834	2287.66	2641	45.3206
8	981	335	646	0.3415	2657.64	1779	51.9403
9	981	460	521	0.4689	1728.26	2144	36.513
10	986	269	717	0.2728	2401.82	1053	61.145

Jenis	PDR	E2E	RO
Modifikasi	0.35499	2510.953	56.47921
Original	0.29084	2680.123	76.9645
Mod - Ori	0.06415	-169.17	-20.48529

B.14 Tabel Hasil Skenario *Node* 160 *node*

MODIFIKASI							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	978	429	549	0.4387	787.84	1888	41.1189
2	980	309	671	0.3153	2155.48	1408	84.6731
3	990	370	620	0.3737	2262.74	1615	53.2432
4	979	253	726	0.2584	3359.86	1185	60.0079
5	982	360	622	0.3666	3439.01	1688	59.0444
6	989	389	600	0.3933	3061.44	1646	46.6787
7	981	335	646	0.3415	2657.64	1779	51.9403
8	997	430	567	0.4313	1991.56	2227	51.8419

9	1007	283	724	0.281	2893.42	1334	60.2261
10	974	341	633	0.3501	2500.54	1514	56.0176

NORMAL							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	975	197	778	0.2021	1815.68	948	115.0558
2	989	399	590	0.4034	1709.8	1738	55.2281
3	979	131	848	0.1338	4721.26	640	122.5802
4	990	265	725	0.2677	1976.99	1332	67.0943
5	973	39	934	0.0401	3069.91	209	271.5385
6	986	192	794	0.1947	4432.21	997	133.2292
7	989	577	412	0.5834	2287.66	2641	45.3206
8	981	335	646	0.3415	2657.64	1779	51.9403
9	981	460	521	0.4689	1728.26	2144	36.513
10	986	269	717	0.2728	2401.82	1053	61.145

Jenis	PDR	E2E	RO
Modifikasi	0.35499	2510.953	56.47921
Original	0.29084	2680.123	95.9645
Mod - Ori	0.06415	-169.17	-39.48529

B.15 Tabel Hasil Skenario *Node* 180 *node*

MODIFIKASI							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	990	228	762	0.2303	5303.99	1117	105.2544
2	981	682	299	0.6952	1088.38	3467	35.8563
3	985	393	592	0.399	2518.39	1834	56.8448
4	996	385	611	0.3865	3200.34	1769	45.1481
5	998	314	684	0.3146	2266.52	1422	72.7197
6	995	621	374	0.6241	1664.65	2729	42.1481
7	993	241	752	0.2427	3436.94	1245	101.9917
8	992	332	660	0.3347	2126.94	1531	70.7229
9	998	314	684	0.3146	2266.52	1422	72.7197
10	995	357	638	0.3588	3489.96	1686	76.1064

NORMAL							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	991	304	687	0.3068	3554.61	1497	68.9934
2	974	256	718	0.2628	3155.72	1310	103.8594
3	994	492	502	0.495	3343.09	2605	45.7764
4	994	214	780	0.2153	4052.26	953	101.8411
5	983	184	799	0.1872	5132.61	852	102.2391

6	981	234	747	0.2385	3835.76	1190	118.8547
7	979	574	405	0.5863	1473.05	2702	44.3937
8	995	156	839	0.1568	3317.92	734	120.2308
9	994	148	846	0.1489	3566.75	635	141.8919
10	985	289	696	0.2934	2932.77	1672	112.2803

Jenis	PDR	E2E	RO
Modifikasi	0.39005	2736.263	67.95121
Original	0.2891	3436.454	96.03608
Mod - Ori	0.10095	-700.191	-28.08487

B.16 Tabel Hasil Skenario *Node* 200 *node*

MODIFIKASI							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	994	368	626	0.3702	2796.14	1641	79.7989
2	983	173	810	0.176	5481.38	818	121.052
3	961	245	716	0.2549	3527.63	1063	86.9959
4	967	153	814	0.1582	3417.34	1320	144.7974
5	982	258	724	0.2627	4378.72	1275	101
6	990	410	580	0.4141	2361.99	2127	97.161
7	971	288	683	0.2966	1881.78	1362	106.9583
8	993	465	528	0.4683	2727.69	2146	77.1742

9	985	155	830	0.1574	3998.57	1718	158.4645
10	972	344	628	0.3539	4869.4	1856	85.5814

NORMAL							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	978	205	773	0.2096	3244.68	981	98.3024
2	971	34	937	0.1446	5011.46	175	140.7857
3	979	127	852	0.1297	2444.37	562	129.6378
4	979	165	814	0.1685	3069.92	840	100.0606
5	980	500	480	0.5102	2752.97	2291	72.572
6	976	263	713	0.2695	3176.71	1332	97.9544
7	978	218	760	0.2229	6119.41	1048	112.5688
8	979	28	951	0.1236	5011.46	175	160.7857
9	971	34	937	0.135	3813.15	174	159.1765
10	984	40	944	0.0407	4197.99	187	370.85

Jenis	PDR	E2E	RO
Modifikasi	0.29123	3544.064	105.89836
Original	0.19543	3884.212	144.26939
Mod - Ori	0.0958	-340.148	-38.37103

B.17 Tabel Hasil Skenario *Speed* 5 m/s

MODIFIKASI							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	975	966	9	0.9908	457.89	4632	7.3561
2	984	806	178	0.8191	898.77	4213	11.6799
3	985	674	311	0.6843	1415.65	3654	15.8872
4	980	969	11	0.9888	549.64	4375	9.8184
5	989	672	317	0.6795	1424.24	3411	16.1071
6	980	885	95	0.9031	785.68	3691	10.8226
7	975	966	9	0.9908	65.03	4307	5.6356
8	1009	834	175	0.8266	698.59	3753	9.7554
9	993	953	40	0.9597	189.65	4348	11.0283
10	999	893	106	0.8939	1299.68	4362	8.71

NORMAL							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	984	808	176	0.8211	925.05	3731	9.8267
2	981	583	398	0.5943	482.95	2589	10.0652
3	974	587	387	0.6027	1096.09	2734	20.6133
4	984	765	219	0.7774	1033.68	3893	8.7817
5	981	716	265	0.7299	689.11	3491	13.838

6	989	888	101	0.8979	388.18	4184	7.6036
7	974	815	159	0.8368	560.17	3562	12.7755
8	988	929	59	0.9403	427.69	3727	10.9666
9	992	889	103	0.8962	839.62	4280	10.1395
10	987	573	414	0.5805	2249.86	3245	20.9389

Jenis	PDR	E2E	RO
Modifikasi	0.87366	778.482	10.68006
Original	0.76771	869.24	12.5549
Mod - Ori	0.10595	-90.758	-1.87484

B.18 Tabel Hasil Skenario *Speed* 20 m/s

MODIFIKASI							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	1001	697	304	0.6963	2381.17	3270	25.5983
2	978	569	409	0.5818	1352.27	2460	18.2425
3	976	619	357	0.6342	555.74	3131	17.0889
4	984	565	419	0.5742	1213.31	2450	19.2071
5	987	423	564	0.4286	1948.4	1624	21.5603
6	986	612	374	0.6207	1577.92	2817	17.915
7	983	588	395	0.5982	2407.11	2654	31.4354
8	979	796	183	0.8131	424.18	3888	14.1357

9	982	669	313	0.6813	993.14	3094	24.7205
10	975	879	96	0.9015	683.97	4192	18.0478

NORMAL							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	982	388	594	0.3951	2416.09	2044	30.0825
2	983	531	452	0.5402	1318.87	2591	17.484
3	992	667	325	0.6724	1401.13	2992	25.7151
4	969	840	129	0.8669	726.8	4102	23.5667
5	989	548	441	0.5541	3913.83	2571	28.1898
6	994	581	413	0.5845	2101.93	2715	33.043
7	980	596	384	0.6082	1825.98	2662	24.1913
8	1000	725	275	0.725	1249.66	3333	26.8083
9	983	385	598	0.3917	3454.85	1777	35.8494
10	978	255	723	0.2607	2591.1	1265	37.9216

Jenis	PDR	E2E	RO
Modifikasi	0.65299	1353.721	20.79515
Original	0.55988	2100.024	28.28517
Mod - Ori	0.09311	-746.303	-7.49002

B.19 Tabel Hasil Skenario *Speed* 40 m/s

MODIFIKASI							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	985	716	269	0.7269	1308.97	3453	28.176
2	982	640	342	0.6517	1848.73	2874	23.6094
3	982	753	229	0.7668	1799.13	3537	23.3705
4	975	794	181	0.8144	1360.16	3707	27.5718
5	984	654	330	0.6646	2169.36	3055	26.2263
6	968	544	424	0.562	1293.54	2554	31.2831
7	987	718	269	0.7275	774.04	3808	24.6964
8	985	689	296	0.6995	2257.9	3180	28.8447
9	977	642	335	0.6571	1980.53	3007	28.2212
10	989	764	225	0.7725	1160.93	3518	21.5052

NORMAL							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	990	403	587	0.4071	2281.94	1951	28.4913
2	978	582	396	0.5951	1305.7	2931	35.2577
3	972	303	669	0.3117	2721.45	1340	38.0726
4	997	523	474	0.5246	1382.27	2476	30.0229
5	980	429	551	0.4378	2329.45	2101	38.62

6	989	398	591	0.4024	1284.06	1935	39.3166
7	980	373	607	0.3806	2896.04	1838	46.8847
8	962	300	662	0.3119	1394.29	1334	44.02
9	989	484	505	0.4894	2448.6	2267	30.0537
10	983	703	280	0.7152	2137.11	3378	23.7923

Jenis	PDR	E2E	RO
Modifikasi	0.7043	1595.329	26.35046
Original	0.45758	2018.091	35.45318
Mod - Ori	0.24672	-422.762	-9.10272

B.20 Tabel Hasil Skenario *Speed* 60 m/s

MODIFIKASI							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	995	610	385	0.6131	1703.03	3001	38.4557
2	974	746	228	0.7659	959.94	4020	34.5228
3	994	731	263	0.7354	992.6	3494	29.368
4	1000	688	312	0.688	1503.63	4297	32.2064
5	992	654	338	0.6593	1508.91	3152	27.5076
6	976	865	111	0.8863	925.14	4098	33.1399
7	996	584	412	0.5863	2600.73	2843	29.9829
8	987	747	240	0.7568	1477.44	3898	29.6386

9	999	634	365	0.6346	1204.97	3213	39.1546
10	980	746	234	0.7612	1605.9	3690	29.0992

NORMAL							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	977	553	424	0.566	917.62	2675	37.4828
2	986	420	566	0.426	1619.59	2288	34.081
3	981	571	410	0.5821	1352.82	3466	41.3765
4	985	634	351	0.6437	2119.11	3350	29.1388
5	986	758	228	0.7688	1383.5	3956	29.7335
6	993	607	386	0.6113	2124.81	3211	35.0115
7	986	420	566	0.434	1783.5	3956	29.7335
8	972	580	392	0.5967	1397.15	2834	26.1172
9	1002	438	564	0.4371	1524.14	2448	52.7306
10	974	696	278	0.7146	1250.77	3748	41.4626

Jenis	PDR	E2E	RO
Modifikasi	0.70869	1448.229	32.30757
Original	0.57803	1547.301	35.6868
Mod - Ori	0.13066	-99.072	-3.37923

B.21 Tabel Hasil Skenario *Speed* 80 m/s

MODIFIKASI							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	999	551	448	0.5516	1540.42	2920	33.1216
2	980	642	338	0.6551	1643.36	3220	33.2897
3	989	689	300	0.6967	1640.11	3812	41.7329
4	980	510	470	0.5204	1149.58	2669	39.5725
5	998	538	460	0.5391	3307.1	2686	38.9257
6	985	684	301	0.6944	1334.99	3269	33.2895
7	1004	580	424	0.5777	1676.52	3389	45.5069
8	982	724	258	0.7373	1751.05	3870	30.2017
9	1003	790	213	0.7876	1577.11	4243	34.1772
10	984	655	329	0.6657	1617.35	3343	37.0321

NORMAL							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	991	539	452	0.5439	2171.04	2769	41.8924
2	976	508	468	0.5205	2277.45	2731	46.3228
3	980	641	339	0.6541	1168.86	3173	42.2465
4	984	604	380	0.6138	1091.27	3077	44.2781
5	981	600	381	0.6116	1474.12	3314	39.9367

6	986	660	326	0.6694	1840.19	3302	38.1576
7	978	803	175	0.8211	1002.45	4350	38.4483
8	990	615	375	0.6212	1664.66	3011	40.361
9	985	397	588	0.403	2549.52	1988	40.6801
10	1006	621	385	0.6173	2834.05	3026	36.9147

Jenis	PDR	E2E	RO
Modifikasi	0.64256	1723.759	36.68498
Original	0.60759	1807.361	40.92382
Mod - Ori	0.03497	-83.602	-4.23884

B.22 Tabel Hasil Skenario *Speed* 100 m/s

MODIFIKASI							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	979	444	535	0.4535	2113.76	2360	53.7973
2	996	499	497	0.501	1328.86	2819	48.8056
3	994	594	400	0.5976	1911.55	2752	36.2054
4	981	685	296	0.6983	868.88	3554	49.1504
5	982	577	405	0.5876	2242.15	3290	44.8735
6	1007	694	313	0.6892	1764.55	3786	38.2824
7	994	767	227	0.7716	2054.56	4029	31.2881
8	993	542	451	0.5458	1908.5	2785	29.3911

9	986	787	199	0.7982	829.02	3948	39.9543
10	992	614	378	0.619	1939.2	3423	42.0195

NORMAL							
No	Packet Send	Packet Receive	Packet Loss	Packet Delivery Ratio	End-to-end Delay	ForwardLine	Routing Overhead
1	969	545	424	0.5624	2095.27	2778	43.8642
2	973	691	282	0.7102	1626.05	3609	40.4052
3	987	465	522	0.4711	2501.29	2282	49.8796
4	984	392	592	0.3984	2110.78	2005	51.6939
5	987	465	592	0.3984	2095.27	2778	49.8796
6	998	491	507	0.492	1584.22	2507	43.3523
7	978	470	508	0.4806	1217.56	2548	49.8894
8	984	670	314	0.6809	1626.05	3909	41.4269
9	982	567	415	0.5774	1500.9	3041	50.9065
10	981	561	420	0.5719	1221.81	2983	49.7398

Jenis	PDR	E2E	RO
Modifikasi	0.62618	1696.103	41.37676
Original	0.53433	1757.92	47.10374
Mod - Ori	0.09185	-61.817	-5.72698

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Maulana Sechan, lahir di Surabaya, 2 April 1997. Penulis adalah anak pertama dari dua bersaudara. Penulis menempuh pendidikan sekolah dasar di SD Negeri No. 274 Surabaya lalu melanjutkan pendidikan sekolah menengah pertama di SMP Negeri 12 Surabaya dan penulis menempuh pendidikan menengah atas di SMA Negeri 18 Surabaya. Selanjutnya penulis melanjutkan pendidikan sarjana di Departemen Teknik Informatika, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember Surabaya. Selama kuliah, penulis aktif dalam berbagai organisasi baik tingkat jurusan maupun universitas.

Dalam menyelesaikan pendidikan S1, penulis mengambil bidang minat Arsitektur dan Jaringan Komputer (AJK). Sebagai mahasiswa, penulis berperan aktif dalam beberapa organisasi kampus seperti fasilitator GERIGI ITS 2016, asisten dirjen Multimedia Kominfo BEM ITS, konseptor GERIGI ITS 2017, staf Kominfo ITS Billiard. Selain itu, penulis juga menjadi staf REEVA SCHEMATICS 2016 dan staf ahli REEVA pada acara SCHEMATICS 2017. Penulis pernah melakukan kerja praktik di Telkomsel Bandung pada Juni – Agustus 2018 dan membuat aplikasi berbasis *web Business Engine Diagnostic*. Penulis dapat dihubungi melalui nomor *handphone*: 081233368216 atau *email*: maulanasechan123@gmail.com